T.Sato / P.Mapstone / I.Muriel

HANDBOEK MSX

Kluwer Technische Boeken

T.Sato
P.Mapstone
I.Muriel
HANDBOEK MSX

MSX

T.Sato / P.Mapstone / I.Muriel

HANDBOEK

MSX

HANDLEIDING VOOR PROGRAMMEURS

KLUWER TECHNISCHE BOEKEN B.V. - DEVENTER - ANTWERPEN

Lay-out: Wim Wijnolts

ISBN 90 201 1876 5
D/1986/0108/207

Oorspronkelijke titel: The Complete MSX, Programmers Guide
Uitgegeven bij: Melbourne House Publishers

1e druk 1986

Ondanks alle aan de samenstelling van de tekst bestede zorg, kan noch de redactie noch de uitgever aansprakelijkheid aanvaarden voor eventuele schade, die zou kunnen voortvloeien uit enige fout, die in deze uitgave zou kunnen voorkomen.

Inhoud

Deel 1
Inleiding tot MSX-BASIC

Hoofdstuk 1

Installeren

Als u uw MSX-computer uitpakt, zult u een video-aansluitkabel aantreffen. Deze kabel heeft aan beide zijden een PHONO-aansluiting en wordt gebruikt om de computer aan een televisietoestel te koppelen.
Op dit moment hoeft u alleen de computer aan een televisietoestel te koppelen. Iedere televisie is hiervoor geschikt. Een kleurentoestel biedt echter het voordeel dat ook de MSX-kleurencommando's kunnen worden gebruikt, maar een zwart/wit-toestel is ook zeer geschikt.
Om de TV op de computer aan te sluiten, steekt u één van de PHONO-pluggen in de achterzijde van de computer met het opschrift 'TV'; het andere einde steekt u in de antenne-aansluiting van uw TV. Heeft uw toestel twee antenne-aansluitingen, 'UHF' en 'VHF', gebruik dan 'UHF'.
Schakel beide apparaten nu in. Als eerste zet de u de TV aan, en laat deze even warm worden, vervolgens schakelt u de computer aan. Als de TV goed is afgestemd, verschijnt eerst het volgende scherm...

```
                MSX system
                version 1.0
        Copyright 1983 by Microsoft
```

... en na een korte pauze het volgende scherm:

```
MSX BASIC version 1.0
Copyright 1983 by Microsoft
28815 Bytes free
OK
■





color      auto      goto      list      run
```

Als u dit tweede scherm niet te zien krijgt, stel dan uw TV bij, totdat het verschijnt. (Het getoonde aantal vrije bytes hangt af van de geheugencapaciteit die op uw MSX-machine is geïnstalleerd.) Dit gebeurt telkens als de MSX wordt aangeschakeld. Als uw TV continu variabele afstemcontrole heeft, stel deze dan bij totdat u een helder beeld heeft. Heeft uw toestel voorkeurtoetsen voor ieder station, dan is het raadzaam een nog vrije toets te gebruiken.

U bent nu zó ver dat u het toetsenbord kunt gaan gebruiken. Oefen zo veel u wilt, u kunt de computer niet beschadigen! Alles wat u intikt, verschijnt op het beeldscherm. Wellicht krijgt u verschillende foutboodschappen, maar maakt u zich geen zorgen, u leert later hoe u deze kunt vermijden.

Wanneer u naar het scherm kijkt, ziet u een wit vierkantje, direct onder het 'OK'-bericht. Dit wordt de tekstcursor genoemd, en geeft aan waar het volgende teken zal worden afgedrukt. Probeer enkele letters in te tikken. U doet er goed aan, uzelf te gewennen aan het gebruik van beide handen gedurende het intikken. Het indrukken van een willekeurige toets heeft tot gevolg dat het teken op het scherm wordt afgedrukt. Tik in:

```
The quick brown fox jumps over the lazy dog.
```

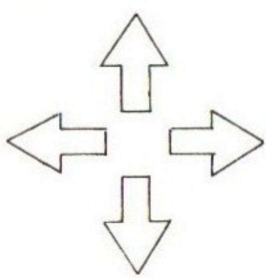
Om spaties te krijgen, drukt u de lange balk in, onderaan het toetsenbord. Om hoofdletters te krijgen, moet u de <SHIFT>-toets tegelijk met de lettertoets indrukken.

Op de linkerzijde van het toetsenbord bevindt zich een toets met het opschrift <CAPS LOCK>. Als u deze toets indrukt, zal alles wat u daarna intikt in hoofdletters worden afgedrukt en het groene lampje er naast zal oplichten. Om hiermee te stoppen, drukt u de toets nogmaals in.

Kijk nu naar de cijfers. Zie wat er gebeurt als u op <SHIFT> drukt en op elk van de cijfers 1 tot en met 9.

Aan de uiterste bovenzijde van het toetsenbord bevinden zich verlengde toetsen met het opschrift F1 tot en met F10. Dit zijn de functietoetsen. Het beeld onderaan het scherm is hier het gevolg van. Vergeet ze nu even,

we komen er in hoofdstuk 8 op terug. Zoek nu de <STOP>-toets op. Deze zult u vaak samen met de <CTRL>-toets gebruiken. De <ESC>-toets voert op dit moment niets uit, maar kan worden gebruikt om een afdrukeenheid te besturen.
De <TAB>-toets verplaatst de cursor op het scherm. Alles wat daarbij wordt gepasseerd, wordt uitgewist. Stond de cursor aanvankelijk links op het scherm, dan wordt deze verplaatst naar de negende positie. Nog eens indrukken van <TAB> beweegt de cursor naar de zeventiende positie, daarna naar de vijfentwintigste.
De <GRAPH>-toets biedt de mogelijkheid de grafische tekens af te drukken die kunnen worden benaderd via de lettertoetsen. Bij het indrukken van <GRAPH> met een willekeurige lettertoets verkrijgt u één van de grafische tekens. Een ander grafisch teken krijgt u als u op <SHIFT>, <GRAPH> en één van de lettertoetsen drukt. Zoek de grafische tekens op die horen bij: harten, schoppen, ruiten en klaveren. Na het indrukken van <CODE> en een lettertoets krijgt u de Europese en Griekse tekens, bijvoorbeeld alpha, bêta, gamma enz. Oefen met deze toetsen. Probeer ook de <SHIFT>-toets in te drukken. De meeste toetsen geven dan een ander CODE-teken.
Een zeer belangrijke toets is de <RETURN>-toets. Elke keer als u de computer wilt laten lezen wat u heeft ingetikt op het beeldscherm, moet u op deze toets drukken. Als u deze nu indrukt, krijgt u wellicht een foutboodschap. Maakt u zich geen zorgen, het betekent dat de computer niet heeft begrepen wat u heeft ingetikt.
Het scherm zal nu wel erg vol zijn. Let op wat er gebeurt zodra het scherm vol raakt; alle tekst beweegt automatisch een regel omhoog, waardoor de bovenste regel verloren gaat. Men noemt dit 'scrollen'.
Om het scherm schoon te maken, moet u tegelijkertijd op <SHIFT> en <HOME> drukken. <HOME> is één van de vier bewerkingstoetsen rechts bovenaan het toetsenbord. Zie hoe de cursor automatisch terugspringt naar de linker bovenkant van het scherm als u dit doet. Als het scherm druk en onoverzichtelijk wordt, kunt u dus op deze wijze het scherm schoonmaken. Drukt u alleen op de <HOME>-toets, dan springt de cursor naar de linker bovenzijde van het scherm, zonder dat daarbij iets wordt uitgewist.
Tik nu een hele regel letters in. Om deze regel te verwijderen, kunt u de spatie-terugtoets indrukken <BS>. Dit heeft tot gevolg dat alles links van de cursor wordt verwijderd. Evenals bij de andere toetsen, herhaalt de toets zich als u hem langere tijd houdt ingedrukt. Uiteindelijk zal daarom de hele regel zijn uitgewist. Tik nog wat meer letters in en oefen met de cursortoetsen die als volgt zijn aangegeven:



Beweeg de cursor in de aangegeven richting zonder dat de te passeren tekst wordt uitgewist.
Probeer een vooraf bepaald teken te verwijderen, door gebruik te maken van de <BS>-toets en de cursortoetsen.
Als u denkt te hebben begrepen hoe een en ander werkt, kijk dan eens wat er gebeurt als u op <DEL> drukt, welke ook één van de vier bewerkingstoetsen is, rechts op het toetsenbord.
Deze toets wist alles uit wat de cursor passeert en verplaatst alles wat rechts van de cursor staat één positie terug. Houdt u <DEL> ingedrukt, dan zal uiteindelijk alles rechts van de cursor worden verwijderd.
Veronderstel dat u het volgende heeft ingetikt:

```
abcfg
```

U zou graag willen zien dat 'd' wordt geplaatst tussen de 'c' en de 'f' in. Om dit te bereiken, moet u de cursor op de 'f' plaatsen en vervolgens op de <INS>-toets drukken. De cursor wordt nu vervangen door een witte regel. Als u nu 'd' intikt, wordt deze geplaatst tussen de 'c' en de 'f' in. Om de cursor terug te krijgen, drukt u nogmaals op <INS>. Oefen het gebruik van de cursortoetsen en de bewerkingstoetsen.
De vierde bewerkingstoets, <SELECT> voert op dit moment niets uit.

Oefeningen

1. Tik de grafische tekens in, die verbonden zijn aan ieder van de lettertoetsen, samen met de bijbehorende letter. Verwijder vervolgens de alfanumerieke tekens, zodat alleen de grafische tekens op het beeldscherm overblijven.
2. Tik zowel het cijfer 0 als de letter O in. Let op het verschil tussen de twee. U moet 0 intikken als het om een cijfer gaat, omdat de computer dit anders weigert. Vergelijk ook het cijfer 1 met de kleine letter l. Ook nu weer oppassen voor verwarring.

Hoofdstuk 2

Commando-mode

Wanneer u eenmaal vertrouwd bent geraakt met het toetsenbord en de <BS>-toets en de bewerkingstoetsen, bent u in staat om de computer opdrachten te geven. Eerst wordt het scherm schoongemaakt door tegelijkertijd op de toetsen <SHIFT> en <CLS> te drukken. Doe dit steeds als het scherm vol raakt en onoverzichtelijk wordt. Tik nu voorzichtig in:

```
PRINT "Welkom bij uw MSX"
```

en druk op de <RETURN>-toets. 'Welkom bij uw MSX' verschijnt direct onder de door u zojuist ingetikte regel, gevolgd door een OK-boodschap. Misschien heeft u een foutboodschap gekregen. Maak u geen zorgen, probeer het nog eens, wellicht heeft u een tikfout gemaakt.
Na het drukken op <RETURN>, voert de computer een opdracht uit. Hij herkent alleen specifieke woorden, zogenaamde sleutelwoorden, waarvan PRINT er één is. Na het lezen van PRINT, gevolgd door de tekst tussen aanhalingstekens, weet de computer dat alles tussen aanhalingsteken moet worden afgedrukt.
De computer accepteert zowel kleine letters als hoofdletters en negeert spaties. (Plaats echter nóóit spaties tussen de letters van een sleutelwoord, omdat de computer dit niet accepteert.)

```
print "Welkom bij uw MSX"     <RETURN>
```

levert hetzelfde resultaat.
Alles tussen aanhalingstekens wordt precies zo afgedrukt als u het heeft ingetikt.
De regel 'Welkom bij uw MSX' wordt string genoemd. Als de computer een PRINT-opdracht leest, gevolgd door een string, verwijdert hij de aanhalingstekens en drukt de rest af zoals het is ingetikt. Later meer over strings!
Om een getal af te drukken, zijn geen aanhalingstekens nodig. Tik in:

```
PRINT 12345          <RETURN>
```

12345 wordt afgedrukt direct onder de laatste regel op het scherm. Ook hier doet het er niet toe hoeveel spaties u tussen PRINT en het getal plaatst. Spaties in het getal worden genegeerd.

```
PRINT 12 345        <RETURN>
```

geeft hetzelfde resultaat.
Als u nu intikt:

```
PRINT 3+4       <RETURN>
```

drukt de computer de waarde 7 af. De uitdrukking 3+4 wordt geëvalueerd en het resultaat wordt u gegeven. Wilt u 3+4 afdrukken zoals het hier staat, dan moet u gebruik maken van een string, met behulp van aanhalingstekens.

```
PRINT "3+4"       <RETURN>
```

De computer drukt af 3+4, want hij herkent het als een string.
Om een waarde af te trekken, gebruikt u '-'.
Om een waarde te vermenigvuldigen, gebruikt u '*', in plaats van 'x'; dit is om verwarring te voorkomen met de letter 'x'.
Om te delen gebruikt u '/'. Tik het volgende achtereenvolgens in. N.B. Vergeet niet na elke regel op <RETURN> te drukken. Dit wordt vanaf nu niet meer apart vermeld.

```
PRINT 4*3
PRINT "3+4=";3+4
```

Het laatste voorbeeld geeft:

```
3+4= 7
```

Als de computer een puntkomma leest, verwacht hij nog meer af te drukken tekst, in dit geval de uitdrukking 3+4.
De computer evalueert de uitdrukking en drukt het resultaat af achter de string. U zult merken dat zowel voor als achter een positief getal een extra spatie wordt afgedrukt.

```
PRINT "aa"  ; 3;4;-5  "bb""cc"
```

geeft:

```
aa 3  4 -5 bbcc
```

Zoals u ziet, blijven er bij de strings geen spaties over. Bij een negatief getal wordt het negatieve teken op de extra spatie geplaatst. Als u geen interpunctie gebruikt, veronderstelt de computer een puntkomma. Bij cijfers moet u wel interpunctie gebruiken, omdat de computer deze anders als een lang cijfer beschouwt.
De computer reageert ook op een komma in een PRINT-opdracht. Dit heeft tot gevolg dat items wel op een regel worden afgedrukt, maar dat het tweede item 15 posities verder dan het eerste wordt afgedrukt.
Tik de volgende regels in:

```
LET A=3
PRINT A
```

Het cijfer 3 wordt getoond op het scherm.
De LET-opdracht kent een waarde toe aan een variabele, in dit geval A. Als de computer een PRINT-opdracht ziet, verwacht hij een cijfer omdat de aanhalingstekens ontbreken. Daarom kijkt hij in zijn geheugen, vindt dat er een waarde is toegekend aan de variabele A, en drukt deze waarde af.
U kunt ook ingewikkelde uitdrukkingen meegeven aan een LET-opdracht:

```
LET B=31*72*4
PRINT B
```

Het getal 8928 wordt getoond.
Tik nu in:

```
LET B=B+1000
PRINT B
```

U krijgt nu:

```
9928
```

De bovenstaande LET-opdracht lijkt wiskundig incorrect, maar is voor de computer zeer acceptabel. De uitdrukking rechts van '=' wordt geëvalueerd en geplaatst in de variabele links van '='. In dit geval is de variabele B gelijk aan de oude waarde, plus 1000.
Namen van variabelen moeten beginnen met een letter. Daarachter mogen cijfers worden gebruikt. Spaties in namen van variabelen worden genegeerd, dus:

A   A   is gelijk aan   AA

Hoofdletter en kleine letters worden geaccepteerd door de computer, doch deze maakt hier geen onderscheid tussen.

ABBACUS   is gelijk aan   abbacus

Het is belangrijk om geen sleutelwoord op te nemen in de naam van een variabele, dus:

HOND

is onjuist omdat ON een sleutelwoord is.
Sommige woorden zijn gereserveerd, en kunnen worden gebruikt in latere versies van de MSX. Deze woorden moeten dus ook niet worden opgenomen in de namen van variabelen, bijvoorbeeld:

NAME

Dit is onjuist, omdat 'NAME' een gereserveerd woord is. Hier volgen enkele toegestane namen van variabelen:

```
nummer2
n2
NUM
```

De volgende namen zijn onjuist:

2n	Het eerste teken moet een letter zijn.
num?	'?' is een onjuist teken in de naam van een variabele. Er mogen geen leestekens voor worden gebruikt.

Ook strings kunnen worden toegekend aan variabelen:

```
LET A$="Hallo"
PRINT A$
```

'Hallo' wordt getoond, omdat PRINT A$ gelijk is aan het intikken van PRINT "Hallo" in dit voorbeeld.
Namen van string-variabelen moeten eindigen met een '$'-teken, om ze onderscheiden van numerieke variabelen. Anders moeten zij voldoen aan de specificaties van namen van numerieke variabelen. Hier volgen enkele correcte namen van string-variabelen:

```
JAAR$
INKOMEN$
L1$
```

De volgende namen zijn incorrect:

A	Geen $-teken.
A.B$	'.' is niet toegestaan (leesteken).
1EERST$	Het eerste teken moet een letter zijn.
ON$	ON is een sleutelwoord.
TOON$	ON is een sleutelwoord.
NAME$	NAME is een gereserveerd woord.

Een andere nuttige opdracht is INPUT. Tik in:

```
INPUT A
```

Wanneer de computer dit leest, toont hij een vraagteken en wacht tot u een getal heeft ingetikt. Dit getal wordt dan toegekend aan de variabele A. Tik een getal in. U ziet het verschijnen na het vraagteken. Als u nu intikt:

```
PRINT A
```

Wordt het door u ingetikte getal afgedrukt. Als u een string intikt,

krijgt u een foutboodschap.

Oefening

1. Bepaal de waarde van A, als:

```
A=49*38.07/13
```

Hoofdstuk 3

Het schrijven van een programma

Tot nu toe heeft u gewerkt in de commando-mode. U heeft een commando ingetikt op het scherm, daarna op <RETURN> gedrukt en de computer heeft dit direct daarna uitgevoerd. Zou u het commando nogmaals willen uitvoeren, dan zou u de regel opnieuw moeten intikken, hetgeen nogal wat tijd kost.
Als u de computer een bepaald commando wilt laten onthouden, dan moet u een regelnummer meegeven, bijvoorbeeld:

```
10 PRINT "DIT IS EEN TEST"
```
<RETURN>

Na het indrukken van <RETURN> leest de computer het regelnummer, in dit geval dus 10, en realiseert zich dan dat deze regel onderdeel is van een programma. Daarom slaat hij de regel op in zijn geheugen. Zodra dit is gebeurd, wordt een 'OK'-bericht afgegeven op het scherm.
Om de computer de regel te laten uitvoeren, gebruikt u het commando RUN. Tik in:

RUN *<RETURN>*

DIT IS EEN TEST wordt nu getoond.
Om nu een programma te schrijven, moet ieder commando worden voorafgegaan door een nummer. Dit nummer kan variëren van 1 tot 65529, maar het is aan te bevelen te beginnen met 10, en telkens te verhogen met 10, zodat u dus achtereenvolgens regelnummers krijgt van 10, 20, 30, 40 enz. Op deze wijze blijft er namelijk ruimte over om regels tussen te voegen, want u kunt gebruik maken van regelnummers als 15, 25, 36 enz. zoals nodig blijkt. De computer voert het programma regel voor regel uit, te beginnen bij de regel met het laagste nummer.
Voordat u het volgende programma intikt, tikt u even in:

NEW *<RETURN>*

Dit commando wist de oude programma's en variabelen, aanwezig in het geheugen van de computer. Het is aan te raden dit commando te gebruiken voordat u aan een nieuw programma begint.
Tik nu het volgende programma in. Vergeet niet na elke regel <RETURN> in te drukken:

```
10 PRINT "Wat is uw naam?"   <RETURN>
20 INPUT N$                  <RETURN>
30 PRINT "HALLO ";N$;" WELKOM BIJ MSX"      <RETURN>
```

RUN dit programma. Indien u geen tikfouten heeft gemaakt, verschijnt het volgende op het scherm:

```
Wat is uw naam?
?
```

De INPUT-opdracht van regel 20 wacht op de invoer van een string-variabele. Tik nu uw naam in en druk op de <RETURN>-toets. De computer drukt dan af:

```
HALLO naam WELKOM BIJ MSX
```

Nadat u uw naam heeft ingetikt, vervolgt de computer met regel 30 en geeft u het welkom-bericht. Als de computer geen regels meer vindt die moeten worden uitgevoerd, weet hij dat het einde van het programma is bereikt en sluit af door de 'OK'-boodschap te geven. U bent dan teruggekeerd in de commando-mode.
Indien u een foutboodschap kreeg in plaats van het welkom-bericht, dan heeft u een tikfout gemaakt in één van de regels. Voordat u op zoek gaat naar de gemaakte fouten doet u er goed aan een nieuwe programmalijst op te vragen. Om deze te verkrijgen tikt u in:

```
LIST
```

Dit commando produceert een keurige lijst van uw programma, te beginnen bij het laagste regelnummer. Alle sleutelwoorden en namen van variabelen worden getoond in hoofdletters, ook al had u ze in kleine letters ingetikt. U zou nu moeten krijgen:

```
10 PRINT "Wat is uw naam?"
20 INPUT N$
30 PRINT "HALLO ";N$;" WELKOM BIJ MSX"
```

Controleer de regel die wordt genoemd in de foutboodschap. Zodra u de fout heeft gevonden, kunt u de regel bewerken door gebruik te maken van de bewerkingstoetsen en cursortoetsen, zoals beschreven in hoofdstuk 1. Als u daarna de cursor heeft geplaatst bij de gecorrigeerde regel, kunt u de computer opdracht geven de oude regel hierdoor te laten vervangen, door te drukken op <RETURN>.
U doet er goed aan de cursor uit de programmalijst te verwijderen, voordat u weer RUN intikt. Om dit te bereiken, houdt u de toets met de pijl die naar omlaag wijst ingedrukt totdat de cursor onderaan het scherm staat. Tik nu RUN weer in, gevolgd door <RETURN>. Herhaal de bewerkingsprocedure totdat uw programma werkt!
Als uw scherm druk en onoverzichtelijk wordt, druk dan op <SHIFT> <CLS>

alvorens LIST in te tikken. Blijkt nu dat u erg veel fouten heeft gemaakt in een bepaalde regel, dan werkt het vaak sneller om de regel helemaal opnieuw in te tikken, in plaats van de cursor- en bewerkingstoetsen te gebruiken. Dus het bewerken van een regel als:

```
10 PRONT WWHST IT US NAAM"
```

doet u het snelst door de regel opnieuw in te tikken:

```
10 PRINT "Hallo, wat is uw naam?"
```

De oude regel 10 wordt dan vervangen door de nieuwe.
Wilt u een bepaalde regel verwijderen uit een programma, dan behoeft u slechts het regelnummer in te tikken, gevolgd door <RETURN>. Nadat u dit heeft gedaan, krijgt u de regel nooit meer terug, dus wees voorzichtig!

```
10 <RETURN>
```

verwijdert regel 10.
Tik het volgende programma eens in:

```
10 REM Dit programma herhaalt zichzelf
20 CLS
30 INPUT "WAT IS UW NAAM ?";N$
40 PRINT "HALLO ";N$;" WELKOM BIJ MSX"
50 GOTO 30
```

De REM-opdracht vertelt de computer dat hij alles kan negeren wat er in deze regel staat. De rest van de regel wordt gebruikt voor commentaar, als geheugensteun. U doet er goed aan REM-opdrachten op te nemen in uw programma, omdat u anders zou kunnen vergeten hoe het programma werkt, als u het later nodig mocht hebben.
De tweede regel maakt het scherm schoon. U weet al dat één van de toetsen dit ook doet, namelijk <SHIFT> <HOME>, maar als u het binnen een programma wilt uitvoeren, moet u gebruik maken van het CLS-commando.
Het commando in regel 30 moet nu bekend zijn. Dit voorbeeld toont hoe u een tekst plaatst voor het vraagteken '?'. Alles wat u moet doen, is de tekst tussen aanhalingstekens plaatsen, daarna een puntkomma plaatsen waarna de naam van de variabele komt. De volgorde is hier belangrijk.
Regel 50 vertelt de computer terug te gaan naar regel 30, en vanaf die regel de uitvoering van het programma voort te zetten. Dit noemt men ook wel een 'oneindige lus', omdat de computer steeds opnieuw terug springt naar regel 30, na het bereiken van regel 50. Dit gaat zo door tot het moment waarop u het programma onderbreekt.
Probeer het programma nu te draaien (RUN). Als u genoeg krijgt van het intikken van uw naam druk dan op <CTRL> <STOP>. U krijgt dan de boodschap:

```
BREAK at LINE 30
```

Om met de uitvoering van het programma door te gaan, tikt u CONT in. U zult zien dat er met de uitvoering wordt doorgegaan vanaf het punt waar het eerder werd onderbroken, namelijk regel 30 bij de INPUT-opdracht.
Had u het programma bij een andere regel onderbroken, bijvoorbeeld een regel zonder INPUT-opdracht, dan zult u zien dat de uitvoering bij de volgende regel begint.

Oefening

1. Schrijf een programma dat naar uw naam en leeftijd vraagt en deze gegevens vervolgens afdrukt.

Hoofdstuk 4

Meer over rekenen

U heeft reeds kennis gemaakt met de wiskundige symbolen:

```
+  -  /  *
```

Een ander nuttig symbool is '↑'. Dit geeft de macht van een getal, bijvoorbeeld:

2^3 wordt geschreven als 2↑3

Tik nu in:

```
PRINT 4↑5
```

Dit levert 1024 op.
Haakjes mogen worden gebruikt in rekenkundige bewerkingen. Iedere uitdrukking tussen haakjes wordt als eerste uitgevoerd.
Alle eerder genoemde symbolen kunnen worden gecombineerd in een uitdrukking. De uitdrukking wordt dan in de onderstaande volgorde geëvalueerd:

( )	Uitdrukkingen tussen haakjes worden als eerste geëvalueerd
↑	Exponenten
*,/	Vermenigvuldigen en delen hebben dezelfde prioriteit
+,-	Deze worden als laatste geëvalueerd

Als voorbeeld evalueert de computer de volgende uitdrukking in drie stappen:

(3+4)*7↑2

eerste stap	7*7↑2
tweede stap	7*49
derde stap	343

Het '*'-symbool moet tussen de haakjes in worden geplaatst, dus:

`PRINT (3+4) (4-2)` is FOUT!

De regel moet als volgt zijn:

```
PRINT (3+4)*(4-2)
```

In het volgende voorbeeldprogramma wordt gevraagd een temperatuur in Fahrenheit te geven die vervolgens wordt omgezet in graden Celsius.

```
10 REM temperatuurconversie
20 CLS
30 INPUT "Temperatuur in Fahrenheit";F
40 C=(F-32)*5/9
50 PRINT "Equivalent in graden Celsius is ";C
60 GOTO 30
```

Merk op dat in regel 40 het LET-commando is weggelaten. Deze opdracht is niet verplicht. Zodra de computer een variabele naast een is-gelijk-teken ziet staan, kent hij de waarde van de uitdrukking rechts van '=' toe aan de numerieke variabele links ervan.
Als u dit programma laat draaien, zult u zien dat de waarde in Celsius in 14 cijfers nauwkeurig wordt gegeven! Wilt u deze waarde echter afronden naar de dichtstbijzijnde integere waarde, dan kunt u de functie INT gebruiken.
INT converteert reële getallen, dit zijn getallen met een decimale punt, naar integere waarden, dus hele getallen. Het argument moet tussen haakjes staan.
De INT-functie rondt naar beneden af, naar de dichtstbijzijnde integere waarde:

3.4 wordt 3
-2.7 wordt -3

Om de reële waarde naar de dichtstbijzijnde integere waarde af te ronden, dus niet alleen omlaag, moet u bij de waarde 0.5 optellen:

INT(3.9) evalueert naar 3
INT(3.9+0.5) evalueert naar 4

U moet zelf de computer laten weten of u de integere of reële waarde wilt bewaren. Om onderscheid te maken tussen deze waarden worden de integere waarden voorzien van het achtervoegsel '%'. Vervang de regels 40 en 50 van het laatste programma door:

```
40 C%=INT((F-32)*5/9+0.5)
50 PRINT "EQUIVALENT in graden Celsius is ";C%
```

Het programma werkt ook als u gebruik maakt van C in plaats van C%, maar het resultaat wordt dan opgeslagen met 14 decimale plaatsen, ook al zijn alle getallen achter de decimale punt nullen. De nullen worden nooit afgedrukt, maar worden wel in het geheugen opgeslagen en verbruiken dus de

kostbare geheugenruimte. Het is daarom beter dat u, waar mogelijk, gebruik maakt van integere namen voor variabelen.

Oefening

1. Pas het temperatuurprogramma zó aan dat u INPUT geeft in graden Celsius en de waarde in Fahrenheit wordt afgedrukt. U zult gebruik moeten maken van de gelijkheid:

   F=C*9/5+32

Hoofdstuk 5

Het gebruik van lussen

Dit programma drukt een willekeurige tafel af. Maak geen probleem van het intikken!

```
10 REM tafel van M
20 INPUT "Vermenigvuldiger";M
30 PRINT "1 maal ";M;"=";1*M
40 PRINT "2 maal ";M;"=";2*M
50 PRINT "3 maal ";M;"=";3*M
```
enz.

Dit wordt een erg lang programma met veel herhalingen. Een zeer beknopte methode om hetzelfde doel te bereiken, is het gebruik van de FOR...NEXT-lus:

```
REM tafel van M met lus
20 INPUT "Vermenigvuldiger";M%
30 FOR C%=1 TO 10
40 PRINT C%;"maal ";M%;"=";C%*M%
50 NEXT C%
60 GOTO 20
```

Let op het gebruik van integere numerieke waarden in dit programma (C% en M%).
De eerste keer dat de computer regel 30 heeft bereikt, zet hij C%=1 en gaat dan door tot regel 40. Als regel 50 wordt bereikt, zorgt de NEXT-opdracht er voor dat de computer teruggaat naar regel 30. Aangekomen bij regel 30, wordt de teller met één verhoogd, dus C% wordt dan 2. Deze lus wordt herhaald totdat C% de waarde 10 heeft bereikt, dit is de grenswaarde; hierna negeert de computer de NEXT-opdracht in regel 50 en gaat met de uitvoering door bij regel 60.
De verhogingsfactor van de teller kunt u vergroten met behulp van de STEP-opdracht in de FOR...NEXT-regel. Probeer het volgende eens:

```
10 FOR N=0 TO 12 STEP 2
20 PRINT N
30 NEXT
```

U zou dan als resultaat te zien moeten krijgen:

```
0
2
4
6
8
10
12
```

De voorwaarden voor de benaming van de teller zijn gelijk aan die voor de namen van numerieke variabelen. Zie ook hoofdstuk 2.
Als u de limiet in bovenstaand programma vervangt door 13 heeft dit geen effect op de resultaten. De computer stopt ook dan met het herhalen van de lus, zodra de waarde 12 is bereikt. Met een STEP-waarde 2 zal nooit 13 kunnen worden bereikt.
U kunt zowel van reële getallen als van integere waarden gebruik maken om de initiële waarde, de grenswaarde en de verhogingsfactor te definiëren. Ook een negatieve STEP is mogelijk:

```
10 FOR N=10 TO 2.5 STEP -2.5
20 PRINT N
30 NEXT N
```

levert op:

```
10
7,5
5
2.5
```

Let u er wél op dat de teller met een hogere waarde begint dan de uiteindelijke grenswaarde, omdat anders een foutboodschap wordt gegeven.
Het is mogelijk om een FOR...NEXT-lus binnenin een andere FOR...NEXT-lus te plaatsen. Dit noemt men een geneste lus. Deze techniek wordt in het volgende voorbeeld toegepast om een patroon van asterisken af te drukken:

```
10 REM *** een geneste lus
20 PRINT "*"
30 FOR I=1 TO 9
40 FOR J=1 TO I
50 PRINT "*";
60 NEXT J
70 PRINT "*"
80 NEXT I
```

U zou dan het sterretjespatroon te zien moeten krijgen zoals op de volgende pagina is afgebeeld.

```
*
**
***
****
*****
******
*******
********
*********
**********
```

Let op de ';' aan het einde van de PRINT-opdracht in regel 50. Dit veroorzaakt dat de volgende PRINT-opdracht op dezelfde regel wordt afgedrukt, op de eerste vrije positie, omdat er geen wagen-terugteken aanwezig is.
Regel 30 zet I=1. Dit bepaalt de limiet van de teller J in de geneste lus, zodat de eerste keer dat de I-lus wordt doorlopen, slechts één keer een '*' wordt afgedrukt. De volgende keer dat de hoofdlus wordt doorlopen, wordt I verhoogd tot 2. J kan nu de waarden 1 en 2 aannemen, dus wordt deze lus tweemaal doorlopen met als resultaat dat er twee '*' worden afgedrukt op dezelfde regel.
Andere patronen zijn mogelijk door het variëren van de STEP-waarde.

Oefeningen

1. Zie wat er gebeurt als u de regels 20 en 70 weglaat.
2. Probeer het volgende programma met een STEP-waarde 4 in de hoofdlus. Verander de grenswaarde van deze lus. Met een veranderde STEP-waarde dienen de regels 20 en 70 te worden aangepast om een regelmatig patroon te behouden.
3. Varieer de STEP-waarde in de geneste lus.

Hoofdstuk 6

Het gebruik van condities

Soms is het nodig dat u uw programma een bepaald pad laat volgen op basis van een bepaalde conditie. Hiervoor wordt het IF...THEN-formaat gebruikt. Hier volgt een voorbeeld van het gebruik ervan:

```
20 IF A=B THEN GOTO 100
30 PRINT A,B
```

Regel 20 vertelt de computer dat, indien (IF) de variabele A gelijk is aan de inhoud van de variabele B, moet worden gesprongen naar (THEN GOTO) regel 100, anders moet worden doorgegaan met de volgende regel van het programma die in dit geval de variabelen A en B afdrukt.
Verschillende uitdrukkingen kunnen volgen op de IF-opdracht met behulp van de volgende tekens:

>	IS GROTER DAN
<	IS KLEINER DAN
=	IS GELIJK AAN
>=	IS GROTER DAN OF GELIJK AAN
<=	IS KLEINER DAN OF GELIJK AAN
<>	IS ONGELIJK AAN

Ieder commando kan volgen op THEN. Bijvoorbeeld:

```
20 IF A=B THEN PRINT A
```

Het volgende programma vraagt om de invoer van twee getallen en drukt deze vervolgens af, het grootste getal eerst.

```
10 INPUT "Getallen";A,B
20 IF A>B THEN GOTO 40
30 SWAP A,B
40 PRINT "Het grootste getal is ";A
50 PRINT "Het kleinste getal is ";B
```

In dit programma worden de twee variabelen ingevoerd met behulp van een INPUT-opdracht. De computer toont 'Getallen?', en wacht tot u twee getallen heeft ingevoerd. U kunt zowel één van de getallen intikken, daarna een

komma, dan het laatste getal invoeren en op <RETURN> drukken, als het eerste getal intikken, dan op <RETURN> drukken, daarna het tweede getal en weer op <RETURN> drukken. Als u de tweede mogelijkheid kiest, toont de computer twee vraagtekens nadat u op <RETURN> heeft gedrukt. Het programma wordt daarna voortgezet.

In het algemeen kunt u elke hoeveelheid variabelen invoeren met het INPUT-commando, zolang ze maar gescheiden zijn van de tekst die tussen aanhalingstekens staat, en de variabelen onderling zijn gescheiden door komma's.

Het SWAP-commando in regel 30 verwisselt de inhoud van variabele A met de inhoud van variabele B. Als dus de conditie A>B in regel 20 niet waar is, wordt in plaats van een sprong naar regel 40, regel 30 uitgevoerd. De SWAP-opdracht zorgt ervoor dat de grootste waarde in de variabele A komt te staan. De volgende twee regels drukken de getallen af. SWAP kan ook worden gebruikt om de inhoud van string-variabelen te verwisselen. De variabelen dienen te worden gescheiden door een komma.

Het IF...THEN-formaat kan worden uitgebreid met het woord ELSE. Alles achter het woord ELSE wordt uitgevoerd, indien de conditie voor THEN niet waar is. Het vorige programma zou als volgt kunnen worden geschreven:

```
10 INPUT "Getallen";A,B
20 IF A>B THEN PRINT "Het grootste getal is";A ELSE PRINT
"Het grootste getal is";B
30 IF A>B THEN PRINT "Het kleinste getal is";B ELSE PRINT
"Het kleinste getal is";A
```

Meerdere condities kunnen worden gebruikt na de IF-opdracht met behulp van AND en OR. Bijvoorbeeld:

```
10 INPUT A,B,C
20 IF A=B AND A>C THEN PRINT A
```

Het laatste korte programma drukt A alleen af, indien A gelijk is aan B en groter is dan C. Vergelijk het bovenstaande met het volgende:

```
10 INPUT A,B,C
20 IF A=B OR A>C THEN PRINT A
```

Deze versie van het programma drukt A af, indien deze gelijk is aan B, of groter is dan C.

De vergelijkingstekens kunnen ook worden toegepast bij strings. Een string wordt in het geheugen van de computer opgeslagen als een getal, waarbij ieder teken een ander getal is. Om strings met elkaar te vergelijken, vergelijkt de computer in feite de twee getallen die ieder de eerste letter van de string voorstellen. Als de eerste twee tekens gelijk zijn, vergelijkt de de computer de volgende twee, enz.

Kleine letters en hoofdletters worden door verschillende getallen voorgesteld:

A - Z door de getallen 65 tot 90
a - z door de getallen 97 tot 122

Andere tekens, zoals '*', kunnen ook worden vergeleken. Hierover later meer (zie hoofdstuk 20)!
De nu volgende strings staan in alfabetische volgorde, de 'grootste' voorop:

"zebra"
"aa"
"aA"
"a"
"Hallo"
"AND"

Om samen te vatten wat u in dit hoofdstuk heeft geleerd, volgt hier een programma dat gebruik maakt van de IF...THEN- en SWAP-opdrachten.
Dit programma sorteert drie opgegeven getallen en toont ze in de gesorteerde volgorde, het grootste getal voorop:

```
10 REM SORTEERPROGRAMMA
20 INPUT "Drie getallen";A,B,C
30 IF B>A THEN SWAP A,B
40 IF C>B THEN SWAP B,C
50 IF A>B THEN SWAP B,A
60 PRINT A,B,C
```

Oefen met verschillende getallen. Kunt u de logica van dit programma bepalen?

Oefeningen

1. Schrijf aan de hand van bovenstaand programma een routine die drie strings op alfabetische volgorde sorteert.
2. Het voorbeeldprogramma werkt slechts met drie getallen. Kunt u een beter programma schrijven dat meer dan drie getallen aankan? Er wordt er één in hoofdstuk 13 beschreven.

Hoofdstuk 7

Nuttige commando's en tips voor het schrijven van programma's

Een nuttig en tijdsbesparend commando is AUTO. Na het intikken hiervan zult u zien dat regelnummer 10 AUTOmatisch op het beeldscherm verschijnt. Tik nu REM in en druk op <RETURN>. Nu verschijnt regelnummer 20. Dit blijft zo doorgaan. Om deze AUTOmatische regelnummering te stoppen, moet u tegelijkertijd <CTRL> en <STOP> indrukken.
Voordat u begint met programmeren is het aan te raden, met behulp van het NEW-commando oude programma's te wissen en het variabelengebied terug te zetten. Als u dit niet doet, kan het gebeuren dat oude programmaregels worden vermengd met het nieuwe programma, vooral als het vorige programma langer was dan het huidige. Dit kan resulteren in een chaos!
Zoals u reeds weet, wordt LIST gebruikt om de programmalijst te tonen. Wenst u alleen één regel te zien van uw programma (als bijvoorbeeld een foutboodschap naar een regel verwijst), tik dan LIST in gevolgd door het regelnummer:

```
LIST 30
```

Nu wordt alleen regel 30 op het scherm getoond.
Na het tonen van de programmalijst is het u wellicht opgevallen dat alle kleine letters van de sleutelwoorden en namen van variabelen zijn vervangen door hoofdletters.
Het is aan te raden om uw programma te voorzien van een titel met behulp van REM en deze opdracht overal in uw programma te gebruiken, waar enig commentaar behulpzaam kan zijn. Dit maakt het leven een stuk eenvoudiger als u toe bent aan het bewerken van uw programma, vooral wanneer u dit enige tijd later doet nadat u het programma oorspronkelijk schreef.
REM-opdrachten kunnen altijd nog worden verwijderd als mocht blijken dat het geheugen vol raakt. Omdat u dus mogelijk een REM-opdracht zou kunnen verwijderen, dient u nooit via een GOTO-opdracht naar een REM-opdracht te verwijzen.
De eenvoudigste manier om een regel uit een programma te verwijderen, is alleen het regelnummer in te tikken en daarna op <RETURN> te drukken. Wilt u echter een groter gedeelte uit het programma verwijderen, dan gaat dit sneller met behulp van de DELETE-opdracht:

```
DELETE 20-80
```

Deze zorgt er voor dat alle regels van 20 tot en met 80 worden verwijderd (inclusief 20 en 80). Doe dit voorzichtig, omdat een eenmaal verwijderde regel alleen maar kan worden terug verkregen door hem helemaal opnieuw in te tikken!
Er mogen meer opdrachten in één regel worden opgenomen, doch de commando's dienen te worden gescheiden door een dubbele punt. Bijvoorbeeld:

```
20 PRINT "a":GOTO 10
```

Dit is een correcte regel. De letter 'a' wordt afgedrukt, en de computer zal naar regel 10 springen. Iedere gewenste hoeveelheid commando's kan op deze wijze aaneen worden gesloten. Dit bespaart geheugenruimte omdat een vaste hoeveelheid geheugenruimte in beslag wordt genomen door de regelnummers zelf. Een programma is echter beter leesbaar, indien u het veelvuldig gebruik van meer commando's in één regel vermijdt. Er bestaat ook een maximum voor het aantal tekens in één regel, namelijk 255 tekens, inclusief spaties. Alles wat meer wordt ingetikt, wordt door de computer genegeerd.
Om dit te illustreren, kunt u het volgende intikken:

```
10 PRINT "***************..."
```

om daarmee het scherm op te vullen met sterretjes (*). Druk nu op <SHIFT> <CLS> en start het programma via RUN; alleen de eerste 247 sterretjes worden getoond. De computer heeft de regel tot en met het 255ste teken uitgevoerd. U ziet dat zodra u de laatste aanhalingstekens die bij een string horen weglaat, de computer er automatisch van uitgaat dat deze er horen. Maak er echter geen gewoonte van, omdat dit tot verwarrende resultaten kan leiden.
Nog iets over PRINT - er bestaat ook een verkorte vorm voor, namelijk het vraagteken. Tik eens in:

```
10 ?
```

Doe nu een LIST. Het vraagteken blijkt te zijn vervangen doot PRINT.
Na controle van uw programma kan blijken dat u één of meer regels bent vergeten. Het is dan mogelijk dat er niet voldoende ruimte bestaat om de regels toe te voegen, vanwege een ongeschikte tussenruimte bij de regelnummering. Mocht dit het geval zijn, dan kunt u gebruik maken van RENUM, hetgeen er voor zorgt dat elk van de regels een nieuw nummer krijgt. Bijvoorbeeld:

```
10 REM
11 REM
12 REM
...
...
```

Om een regel tussen 10 en 11 te plaatsen, tikt u RENUM in. Doe vervolgens

een LIST, en u krijgt te zien:

```
10 REM
20 REM
30 REM
...
...
```

Het is nu eenvoudig geworden om nieuwe regels tussen 10 en 20 te plaatsen, door de nummers 11 en 19 te gebruiken.
Het is mogelijk meer dan één programma in het geheugen op te slaan. Om dit te bereiken, kunt u het eerste programma bewaren tussen, bijvoorbeeld, de regelnummers 10 tot 100, en het tweede tussen 500 en 1000. Als u nu RUN intikt, moet u de computer ervan weerhouden het tweede programma direct na het eerste uit te voeren. Dit doet u via de END-opdracht. Dit is de allerlaatste opdracht van het eerste programma. Als de computer END leest, stopt hij met de uitvoering en zet u terug in de commando-mode.
Om de uitvoering voort te zetten, hetgeen in dit geval het starten van het tweede programma betekent, kunt u gebruik maken van CONT gevolgd door <RETURN>. Ook kunt u het tweede programma direct starten door het volgende te doen:

```
RUN 500
```

Dit start de uitvoering bij regel 500. Hetzelfde resultaat verkrijgt u door in te tikken:

```
GOTO 500
```

Ook de STOP-opdracht kan in een programma worden gebruikt. Deze opdracht, evenals END, zet u terug in de commando-mode. STOP echter produceert een foutboodschap 'BREAK in LINE ...', verwijzend naar het regelnummer met STOP. Na de STOP-onderbreking, omdat u terug bent in de commando-mode, kunt u de uitvoering voortzetten door CONT in te tikken.
U kunt een actief programma onderbreken door <STOP> in te drukken. Om door te gaan met de uitvoering dient u weer <STOP> in te drukken. U kunt nu geen gebruik maken van CONT, omdat dit alleen mogelijk is bij programma's voorzien van een STOP-opdracht. Als u <CTRL> <STOP> indrukt, wordt het programma onderbroken en wordt u in de commando-mode gezet. Eenmaal in de commando-mode aangekomen, kunt u het programma starten vanaf de volgende regel door gebruik te maken van CONT. Tik het volgende korte programma eens in:

```
10 REM
20 PRINT "Goedendag"
30 GOTO 20
```

Start het programma met RUN. Druk nu op <STOP>. Dit onderbreekt het programma. De enige manier om het programma te herstarten, is nogmaals

<STOP> in te drukken. Start het programma opnieuw en onderbreek het deze keer met behulp van <CTRL> <STOP>. Herstart het programma met CONT.

```
10 REM
20 PRINT "HALLO"
30 STOP
40 PRINT "GOEDENDAG"
50 END
60 GOTO 20
```

Herstart dit programma via CONT, zowel na de END- als de STOP-opdrachten. Een nogal gecompliceerde opdracht, die u nog niet nodig heeft, is CLEAR. Deze opdracht vertelt de computer alle aanwezige variabelen in het geheugen te vergeten. CLEAR kan ook worden gebruikt om ruimte te reserveren voor een variabele. Na het inschakelen van de computer kunnen de variabelen maximaal 200 tekens lang zijn. Dit kunt u verhogen tot 255 tekens door in te tikken:

```
CLEAR 255
```

Een ander commando dat betrekking heeft op het geheugen van de computer is FRE. Dit vertelt u hoeveel geheugenruimte nog vrij is in het programmageheugen of het stringgeheugen:

PRINT FRE (0)	geeft de geheugenruimte die over is in het BASIC-programmagebied.
PRINT FRE ("")	geeft de geheugenruimte die over is in het stringgebied.

Ten slotte twee commando's die zijn ontworpen om u behulpzaam te zijn bij het traceren van fouten in uw programma.
TRON, wat staat voor TRACE ON, zorgt er voor dat de computer het regelnummer toont van iedere uitgevoerde regel. Tik eens in:

```
10 REM gebruik van TRON
20 PRINT "Hallo, bent u nog wakker ?"
30 INPUT "Ja of Nee";A$
40 IF A$="Nee" THEN BEEP:GOTO 20
50 PRINT "MOOI"
```

Tik vervolgens TRON in en start het programma. U zou nu het volgende beeld moeten krijgen:

```
[10][20] Hallo, bent u nog wakker ?
[30]     Ja of Nee?
```

Nu wacht het programma op uw invoer. Als u Ja invoert, krijgt u:

```
[40][50] MOOI
```

Als u Nee invoert, krijgt u:

```
[40][20]  Hallo, bent u nog wakker ?
[30]      Ja of Nee?
```

Het BEEP-commando in regel 40 is de eenvoudigste manier om geluid te produceren uit uw computer. Om het traceer-proces te stoppen, kunt u TROFF gebruiken.

Hoofdstuk 8

De functietoetsen

Dit zijn de vijf toetsen bovenaan het toetsenbord. Elk van die toetsen heeft twee functies, waarvan één de toets zelf is, en de ander wordt verkregen door ook <SHIFT> in te drukken. Dus de functies F1, F2, F3, F4 en F5 verkrijgt u door op de desbetreffende toets te drukken en de functies F6, F7, F8, F9 en F10 worden verkregen door tegelijkertijd <SHIFT> en de bijbehorende functietoets in te drukken.
Na het inschakelen van de computer zijn de toetsen reeds voorgeprogrammeerd voor bepaalde functies. Een overzicht van deze functies kunt u zien met behulp van het volgende commando:

```
KEY LIST
```

De inhoud van de eerste vijf toetsen wordt getoond op de laatste regel van het beeldscherm. De andere vijf ziet u door op <SHIFT> te drukken. Om dit beeld te verwijderen, moet u KEY OFF intikken en om het terug te krijgen KEY ON. De laatste drie commando's kunt u meeprogrammeren in uw programma's, indien gewenst.
Op dit moment vergeten we even F1, F6 en F7. Over deze toetsen volgt later meer. De functies van de andere zeven toetsen zijn bij u reeds bekend, en u zult ze wellicht handig vinden:

TOETS	BEELD	Omschrijving
F1	color	Zie hoofdstuk 22.
F2	auto	Deze is geprogrammeerd om AUTO op het beeld te tonen, gevolgd door een spatie, dit is equivalent aan het intikken van 'AUTO'.
F3	goto	Het indrukken van deze toets is equivalent aan het intikken van 'GOTO' gevolgd door een spatie.
F4	list	Dit is equivalent aan het intikken van LIST gevolgd door een spatie.
F5	run	De functie van deze toets wijkt iets af van de reeds genoemde toetsen. Deze is equivalent aan het intikken van RUN gevolgd door <RETURN>, zodat deze toets een programma start, indien aanwezig in het geheugen, direct na het intikken ervan.

F6	color 15,4,7	Zie hoofdstuk 22.
F7	cload"	Zie hoofdstuk 11.
F8	cont	Dit is equivalent aan CONT, gevolgd door <RETURN>. Heeft u een programma onderbroken door op <CTRL> <STOP> te drukken, of door een END- of STOP-opdracht in een programma, dan kunt u het programma herstarten vanaf de volgende regel door <SHIFT> en F8 in te drukken.
F9	list	F9 zal de programmalijst direct afdrukken, vanaf het laatst ingetikte regelnummer en de cursor automatisch aan het begin van de regel plaatsen.

Het is ook mogelijk uw eigen functie onder één van de functietoetsen te brengen met behulp van het commando KEY, gevolgd door een toetsnummer, een komma en ten slotte een string die de nieuwe functie bevat. Tik het volgende in:

```
KEY 1,"PRINT"          <RETURN>
```

Dit heeft tot gevolg dat de opdracht PRINT op het scherm wordt getoond, zodra toets F1 wordt ingedrukt.
Let er op dat ook de lijst onderaan het scherm is aangepast. COLOR is vervangen door PRINT.

```
KEY 6,"NEW"+CHR$(13)
```

programmeert toets F6 zó dat deze een NEW uitvoert. CHR$(13) is equivalent aan het indrukken van <RETURN> (zie hoofdstuk 20 voor uitvoeriger details). Bij het definiëren van een toets mag u een beperkt aantal tekens in de string plaatsen, namelijk 15. Let op dat CHR$(13) slechts één teken voorstelt.

Oefening

1. Programmeer toets F7 zodat deze een programma hernummert, zodra F7 wordt ingedrukt.

Hoofdstuk 9

Meer over PRINT en het beeldscherm

Tot nu toe hebben we ons bezig gehouden met PRINT-opdrachten om strings en variabelen af te drukken met behulp van een komma of puntkomma om de afzonderlijke items van elkaar te onderscheiden. De puntkomma zorgt ervoor dat de items naast elkaar worden afgedrukt; de komma veroorzaakt dat de items op dezelfde regel worden afgedrukt, het eerste item aan het begin van de regel en het tweede vanaf positie 15.
Een puntkomma aan het einde van een PRINT-opdracht onderdrukt het wagenterug-symbool, waardoor een volgende PRINT-opdracht met afdrukken op dezelfde regel begint:

```
10 PRINT "To be, or not to be - ";
20 PRINT "That is the question;"
```

Dit wordt allemaal op één regel afgedrukt:

```
To be or not to be - that is the question;
```

Soms zou het beter uitkomen als u een tabel vol resultaten, of misschien een lijst van mogelijkheden, keurig getabuleerd kon afdrukken. Hiervoor bestaan twee commando's, namelijk TAB en LOCATE.
Om te begrijpen hoe u deze commando's moet gebruiken, dient u te weten welke de verschillende scherm-modes zijn. Er bestaan twee tekst-schermen, SCREEN 0 en SCREEN 1. Na het inschakelen van de computer bevindt u zich automatisch in één van deze schermen. Voor dit hoofdstuk wordt er van uitgegaan dat het standaardscherm SCREEN 1 is. Om er zeker van te zijn dat u zich in dit scherm bevindt, tikt u in:

```
SCREEN 1
```

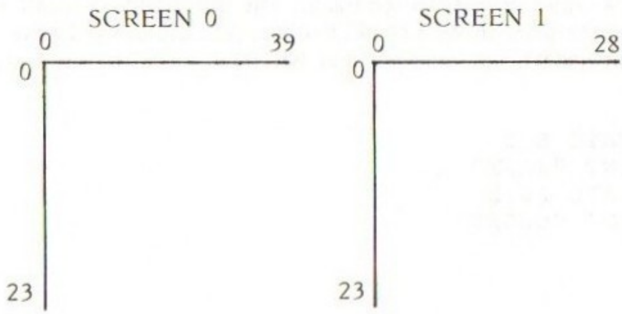
Dit scherm is 29 tekens breed en 24 regels lang. Elk teken neemt een vierkant in beslag bestaande uit 8x8 punten.
Het andere tekstscherm, SCREEN 0, is 39 tekens breed en 24 regels lang. De tekenposities in dit scherm zijn iets kleiner, ieder namelijk 6x8 punten. In dit scherm heeft de schermrand dezelfde kleur als de achtergrond, dus is het scherm helemaal blauw na het inschakelen. Om toegang tot dit scherm te verkrijgen, tikt u in:

```
SCREEN 0
```

Keer nu terug naar scherm 1, door in te tikken:

```
SCREEN 1
```

Het beginpunt van beide schermen bevindt zich links bovenaan het scherm. Let op dat de eerste rij en de eerste kolom rij 0 en kolom 0 worden genoemd. De laatste kolom van SCREEN 0 is daarom kolom 38 en van SCREEN 1 kolom 28. De laatste rij van beide schermen is rij 23.
U kunt niet verder op de laatste regel afdrukken dan nadat u KEY OFF heeft gebruikt om de functietoetsenlijst te verwijderen.



U kunt het maximum aantal tekens per regel in vervangingswaarde laten variëren met behulp van de WIDTH-opdracht, gevolgd door een getal. Dit getal definieert het maximum aantal tekens per regel. In SCREEN 0 kan dit getal maximaal 40 worden, in SCREEN 1 is dit 32. U kunt deze breedte omlaag helpen tot maar liefst één teken als u dat wilt!

```
SCREEN 0
WIDTH 20
```

Deze commando's zetten de maximumbreedte van SCREEN 0 op 20 tekens. U zult zien dat u alleen in de middelste 20 kolommen kunt afdrukken. Om terug te keren naar vervangingsbreedte en -scherm, tikt u in:

```
SCREEN 0
WIDTH 29 .
```

Nu terug naar tabuleren! De TAB-opdracht wordt altijd samen met PRINT gebruikt. Deze opdracht laat de computer weten dat met afdrukken moet worden begonnen bij de opgegeven kolom (of X-coördinaat), gedefinieerd in de TAB-opdracht. Bijvoorbeeld:

```
PRINT TAB(10) "Dit begint met afdrukken in de 11e kolom"
```

Vergeet niet dat de eerste tekenpositie op het scherm bij kolom 0 en rij 0 begint. U zou dit kunnen beschouwen als een X,Y-coördinatenpaar (0,0). U kunt meer TAB-opdrachten in één PRINT-opdracht plaatsen:

```
PRINT TAB(3)4 TAB(7)8
```

Deze opdracht drukt cijfer 4 af in de vierde kolom. De spatie die altijd voorafgaat aan een positief getal wordt in de derde kolom geplaatst. Het cijfer 8 wordt in kolom acht afgedrukt, de spatie in de zevende kolom. Er is geen interpunctie nodig tussen de TAB-opdrachten, maar denk wel aan de haakjes.
U kunt alleen de kolom opgeven in een TAB-opdracht, dus wanneer u in een bepaalde kolom een rij wilt afdrukken, moet u gebruik maken van de LOCATE-opdracht, vlak voor de PRINT-opdracht. Dit commando bepaalt de cursorpositie; het eerste getal in de LOCATE-opdracht stelt het kolomnummer voor (of de X-coördinaat), het tweede getal het rijnummer (of de Y-coördinaat). Bijvoorbeeld:

```
10 LOCATE 6,2
20 PRINT "NAAM"
30 LOCATE 20,2
40 PRINT "SCORE"
```

Dit drukt af:

```
NAAM          SCORE
```

op de derde regel van het scherm. U heeft voor iedere PRINT-opdracht een LOCATE nodig.
Op een andere manier kunt u LOCATE gebruiken om de rij te zetten en vervolgens TAB gebruiken naast PRINT. Het volgende programma geeft hetzelfde resultaat als het vorige:

```
10 CLS
20 LOCATE ,2
30 PRINT TAB(6) "NAAM" TAB(20) "SCORE"
```

Wanneer de X-coördinaat wordt weggelaten in een LOCATE-opdracht veronderstelt de computer dat u wenst te blijven bij de huidige X-positie. Het CLS-commando in regel 10 heeft X=0 gezet, dus wordt deze waarde gebruikt voor het bepalen van de cursorpositie in regel 20. Let op dat ook al laat u de X-coördinaat weg, u de komma niet moet weglaten.
LOCATE mag niet in de commando-mode worden gebruikt, omdat zodra dit commando wordt uitgevoerd, de cursor op de volgende regel wordt geplaatst aan de linkerzijde van het scherm, gereed om een volgend commando in te tikken.
U kunt LOCATE ook gebruiken om de cursor uit te schakelen. Om dit te bereiken, plaatst u een komma achter de Y-coördinaat gevolgd door een nul. Om de cursor terug te krijgen, vervangt u de nul door een één. Bijvoor-

beeld:

```
10 LOCATE ,,0
```

schakelt de komma uit, terwijl:

```
10 LOCATE 10,10,1
```

de cursor weer activeert en deze beweegt naar de positie (10,10). Dit effect kunt u niet zien in de commando-mode.
Nu een ander commando, namelijk SPC. Dit commando wordt binnen een PRINT-opdracht gebruikt om het aantal spaties te definiëren dat moet worden afgedrukt. Het korte programma dat u eerder heeft ingetikt om de titelregel "NAAM   SCORE" af te drukken, zou vervangen kunnen worden door:

```
10 CLS
20 LOCATE ,2
30 PRINTTAB(6)"NAAM"SPC(10)"SCORE"
```

Iedere integere waarde tussen 1 en 255 mag volgen op SPC; het getal moet tussen haakjes staan. Steeds wanneer het nodig blijkt om een string met veel spaties af te drukken, is het aan te raden om gebruik te maken van het SPC-commando, omdat dit geheugenruimte bespaart.
Er is een alternatieve methode om spaties af te drukken door gebruik te maken van SPACE$. Dit is een veelzijdiger commando dan SPC omdat het ook zonder PRINT-opdracht kan worden gebruikt (SPC niet). SPACE$ wordt gebruikt om een string te creëren die alleen uit spaties bestaat. Bijvoorbeeld:

```
A$=SPACE$(10)
```

is gelijk aan:

```
A$="          "
```

Het gebruik van dit commando stelt ons in staat de titelregel aan een variabele toe te kennen en deze net zo vaak af te drukken als we willen:

```
10 H$="NAAM"+SPACE$(10)+"SCORE"
20 CLS
30 LOCATE
40 PRINTTAB(5)h$
```

Denk er aan dat dit niet kan met SPC:

```
A=SPC(10)          IS FOUT !
```

Oefening

1. Schrijf een programma dat om de datum van vandaag en uw geboortedatum vraagt en vervolgens uw leeftijd afdrukt. Tabuleer de berichten op het scherm met behulp van LOCATE. Schakel de cursor uit vlak voor de PRINT-opdracht.

Hoofdstuk 10

Interactief programmeren

Een programma wordt 'interactief' genoemd zodra tijdens de uitvoering ervan om invoer vanaf het toetsenbord wordt gevraagd. Dit hoofdstuk besteedt daarom aandacht aan de INPUT-opdracht en aan INPUT$ en INKEY$. Ondanks het feit dat TAB alleen met PRINT kan worden gebruikt, kan LOCATE daarentegen wel samen met INPUT worden gebruikt:

```
10 LOCATE (10,22)
20 INPUT "Artikel, prijs";A$,p
```

U zult zien dat

```
Artikel, prijs?
```

op rij 22 van het scherm verschijnt. De computer blijft wachten tot u een string en daarna een getal heeft ingevoerd. Dit bereikt u, zoals u reeds heeft gezien, door na het invoeren van de string op <RETURN> te drukken en daarna het getal in te voeren, of door beide elementen, gescheiden door een komma, in één keer in te voeren. Indien u gebruik maakt van de opdracht LINE INPUT, zou u echter een hele regel tekst in kunnen voeren, in een string-variabele:

```
10 LOCATE (10,22)
20 LINE INPUT "Artikelen en prijs?";L$
```

U krijgt te zien:

```
Artikelen en prijs?
```

op de 22ste rij van het scherm. Als u nu intikt:

```
Bananen, pinda's en theezakjes  FL.25,-
```

wordt de gehele regel, inclusief de spaties, in de variabele L$ geplaatst. Als u LINE INPUT vervangt door INPUT, wordt alles achter de komma genegeerd.

LINE INPUT kunt u alleen met string-variabelen gebruiken, dus:

```
10 LINE INPUT A                 IS FOUT!
```

De regel die u intikt, kan een maximum lengte hebben van 200 tekens. De maximale regellengte kan worden vergroot tot 255 posities, met behulp van de CLEAR-opdracht (zie hoofdstuk 7).
In tegenstelling tot INPUT, genereert LINE INPUT niet vanzelf een vraagteken op het scherm, dus moet u deze meegeven aan het einde van het bericht dat volgt op het sleutelwoord. Meerdere regels kunnen worden ingevoerd met een LINE INPUT, maar de string-variabelen dienen dan van elkaar gescheiden te zijn door komma's, en van het bericht, indien aanwezig, door een puntkomma.
Het zou kunnen gebeuren dat het op een bepaalde plaats in uw programma nodig is om te stoppen en een menu van mogelijkheden te tonen van waaruit u een keuze kunt maken. In een spelprogramma kan het bijvoorbeeld gebeuren dat de computer de speler vraagt om de instructies te tonen, of wanneer de speler een spel heeft beëindigd, de computer hem vraagt nog eens te spelen.
U kunt hierbij gebruik maken van de INKEY$-opdracht. Deze functie controleert of er iets op het toetsenbord wordt ingetikt. Is dit het geval dan wordt het bijbehorende teken in de string INKEY$ geplaatst en gaat de computer verder met het uitvoeren van de volgende regel.
Het volgende programmadeel zou geplaatst kunnen worden aan het einde van een spelprogramma:

```
100 CLS
110 LOCATE ,10
120 PRINT "Wilt u nog eens spelen?"
130 PRINT "Toets J voor Ja"
140 PRINT "Toets N voor Nee"
150 A$=INKEY$
160 IF A$="N" OR A$="n" THEN END
170 IF A$="J" OR A$="j" GOTO 10
180 GOTO 150
```

De eerste keer dat regel 150 wordt bereikt, bevat INKEY$ óf een lege string óf een teken behorend bij de ingetikte toets. Indien één van de toetsen 'N', 'n', 'j' of 'J' werd ingedrukt, op het moment dat de computer bij regel 150 is, dan stopt het programma of begint opnieuw bij regel 10. De GOTO-opdracht in regel 180 laat de computer zó vaak terugspringen naar regel 150 tot één van de herkenbare tekens wordt ingedrukt.
Vergeet niet dat INKEY$ een string is, en dus alleen met een string-variabele kan worden vergeleken, dus:

```
A=INKEY$                IS FOUT!
```

en resulteert in de boodschap 'Type Mismatch Error'.
Een vergelijkbare opdracht van INKEY$ is INPUT$. Evenals INKEY$ leest deze het toetsenbord, maar in plaats van het lezen van een teken leest ze het

aantal tekens, zoals meegegeven in de opdracht, dus:

```
10 A$=INPUT$(5)
20 PRINT A$
```

De computer blijft dus wachten tot vijf tekens zijn ingetikt. Regel 20 drukt de tekens vervolgens af. Het getal dat volgt op INPUT$ moet een integer zijn variërend tussen 1 en 200. Gebruikt u een reëel getal dan wordt dit afgebroken tot het dichtstbijgelegen lagere integere getal.

Oefening

1. Gebruik de INPUT$-opdracht om aan het begin van een programma een toegangswoord (wachtwoord) te vragen. De eerste regels van het programma zouden moeten vragen om het toegangswoord, en de uitvoering van het programma mag alleen worden voortgezet als het toegangswoord correct is.

Hoofdstuk 11

Het wegschrijven van uw programma op tape

U bent nu zover dat u redelijk lange programma's kunt schrijven. In plaats van het steeds opnieuw intikken zodra u een programma wilt draaien, is het veel gemakkelijker om het programma op tape te bewaren.
Allereerst moet u de cassetterecorder aansluiten aan uw computer. De meeste cassetterecorders zijn hier geschikt voor; een goedkope MONO is ook goed. Het is zeer handig om een teller te hebben, zodat u de plaats van een bepaald programma op de tape snel kunt terugvinden. De recorder moet een ingang hebben voor een microfoon en een uitgang voor het gebruik van een oortelefoon. Deze contacten moeten 3,5 mm zijn vanwege de stekkers die bij de computer zitten.
De aansluitkabel moet aan één einde een 8 pins DIN-plug hebben. Indien de kabel niet bij uw computer is meegeleverd, doet u er goed aan deze toch aan te schaffen. Zeg dat het voor een MSX-computer is. Steek deze plug vervolgens in de computer, in de ingang met de aanduiding 'CASSETTE'. Het andere einde van de kabel is in drieën gesplitst, met een rode, zwarte en witte stekker.
Steek de witte kabel in de ingang met de aanduiding 'EAR' op uw recorder en de rode stekker in de ingang met 'MIC'. Vergeet de zwarte stekker nog even.
Plaats een lege cassette in uw cassetterecorder. Spoel de tape voldoende terug en speel de tape af tot het einde van het transparante bandje is bereikt dat altijd voorafgaat aan de tape. Zet de tapeteller op 000.
Tik nu een kort programma in. Tik daarna in:

```
CSAVE "NAAM"          <RETURN>
```

U kunt uw programma elke naam geven die u schikt, als deze maar niet langer wordt dan zes tekens. De tekens mogen zowel hoofdletters als kleine letters zijn, of cijfers. Het eerste teken moet een letter zijn.
Druk nu tegelijkertijd de RECORD- en PLAY-toetsen in van uw cassetterecorder en druk pas daarna de <RETURN>-toets in van uw computer.
Zodra de computer klaar is met het wegschrijven van het programma verschijnt een 'OK'-bericht. Stop de tape nu en let op de tapeteller.
Om te controleren of het programma correct is opgeslagen op tape moet u de computer uitschakelen, om zo het nog aanwezige programma uit het geheugen te wissen. Spoel de tape terug tot de teller 000 aangeeft. Schakel de

computer weer aan en tik in:

```
CLOAD "NAAM"    <RETURN>
```

of alleen:

```
CLOAD           <RETURN>
```

Als u de programmanaam weglaat, laadt de computer het eerste programma dat hij tegenkomt op tape. Druk nu op de PLAY-toets van de cassetterecorder. De volgende boodschap verschijnt op het scherm, wanneer de computer het programma heeft gevonden:

```
FOUND: NAAM
```

Zodra het programma is geladen, verschijnt de 'OK'-boodschap weer. Als dit is gebeurd, stopt u de tape.
Let erop dat functietoets F7 is geprogrammeerd om CLOAD" op het scherm af te drukken. Indien u hiervan gebruik maakt, behoeft u alleen nog maar de naam en de laatste aanhalingstekens in te tikken.
U kunt verifiëren of het programma in het computergeheugen gelijk is aan de kopie op tape, door gebruik te maken van CLOAD?. Om hiervan gebruik te maken, spoelt u de tape terug tot aan het begin van het programma en tikt vervolgens in:

```
CLOAD"NAAM"           <RETURN>
```

Start nu de tape. Als het programma in het geheugen gelijk is aan het programma op tape, verschijnt de 'OK'-melding. Krijgt u deze niet te zien dan is het programma fout geladen - zie hiervoor ook de probleemomschrijvingen aan het einde van dit hoofdstuk.
Indien u meer programma's op één tape bewaart, dan doet u er goed aan voldoende ruimte open te laten tussen de programma's en het begin van ieder programma te noteren. Als u een programma over een ander programma heen wegschrijft, is het duidelijk dat het eerste programma verloren is gegaan.
Stel dat u twee programma's op tape heeft bewaard, de eerste met de naam 'EEN' en de tweede met de naam 'TWEE'. Als u nu intikt:

```
CLOAD"TWEE"
```

en de tape vanaf het begin laat draaien, dan krijgt u het volgende beeld te zien:

```
CLOAD"TWEE"
Skip:EEN
Found:TWEE
Ok
```

De computer laat u weten dat hij het programma met de naam 'EEN' heeft

overgeslagen en daarna 'TWEE' heeft geladen. Nu nog de functie van de zwarte stekker. Deze stekker kunt u alleen gebruiken als uw cassetterecorder een REMOTE CONTROL-aansluiting heeft. Is dit het geval, sluit de stekker dan aan en tik in:

MOTOR ON *<RETURN>*

Als u nu een CLOAD/CLOAD? of een CSAVE doet, wordt de motor van de cassetterecorder automatisch uitgeschakeld door de computer als hij klaar is met lezen van of schrijven naar de cassetterecorder. Dit is zeer handig, omdat nogal eens vergeten wordt de cassetterecorder af te zetten na een CLOAD of CSAVE.
Om de REMOTE CONTROL van de cassetterecorder te stoppen, tikt u in:

MOTOR OFF *<RETURN>*

Als u alleen:

MOTOR

intikt, heeft dit hetzelfde effect, omdat dit commando de REMOTE CONTROL uitschakelt als deze niet actief is.

Schrijf- en leesfouten

Heeft u problemen met het laden en bewaren van programma's, controleer dan:

1. Of u de cassetterecorder heeft aangesloten!
2. Of u de kabels correct heeft aangesloten en of ze niet los zijn gaan zitten.
3. Varieer het volume en het geluid van de cassetterecorder. Het is gewoonlijk het beste om het geluid maximaal te zetten en alleen het volume te variëren. Start vanaf de middelste positie en probeer het volume te verhogen of te verlagen.
4. Controleer of u de naam van het programma correct heeft gespeld als een op tape aanwezig programma niet kan worden geladen.

Wanneer u nog steeds lees- en schrijfproblemen heeft, dan heeft u waarschijnlijk een onbetrouwbare cassetterecorder.

Hoofdstuk 12

Het inlezen van gegevens in arrays

Tot nu toe heeft u een beperkt aantal gegevens per keer ingevoerd. Het zal zeker een keer voorkomen dat u een grote hoeveelheid gegevens wilt invoeren, zoals bijvoorbeeld een lange lijst namen of nummers. Stel dat u van plan bent om een telefoonlijst van al uw kennissen bij te gaan houden. Dit zou u als volgt kunnen doen:

```
10 REM TELEFOONBOEK VERSIE 1
20 INPUT "NAAM en NUMMER";N1$,NUM1
30 INPUT "NAAM en NUMMER";N2$,NUM2
40 ...
```

enzovoort, maar bij slechts tien namen wordt dit al een zeer lang programma waarin dezelfde zaken steeds weer worden herhaald. Het zou al veel beter zijn om er een lus in te bouwen voor iedere naam met nummer. Om dit te bereiken, heeft u een zogenaamde array nodig.
Een array is een groep variabelen die alle dezelfde naam hebben, maar van elkaar worden onderscheiden door een vooraf toegekend nummer.
A(n) is een numerieke array. Als n=10 beschikt deze array over elf variabelen met de namen A(0), A(1), A(2), ..., A(10). U zou bij A kunnen denken aan een straatnaam, waarbij het erop volgende nummer een huisnummer voorstelt in die straat.
Als u gebruik wilt maken van arrays met meer dan elf elementen, moet u de computer laten weten dat u deze nodig heeft door de naam en de grootte op te geven. Dit doet u met de volgende opdracht:

```
DIM NUM(12)
```

Deze opdracht reserveert een ééndimensionale numerieke array met de naam NUM die over 13 elementen beschikt, NUM(0), NUM(1), ..., NUM(12). De array wordt ééndimensionaal genoemd omdat hij kan worden voorgesteld door een ééndimensionaal diagram:

```
NUM(0)
NUM(1)
NUM(2)
NUM(3)
```

```
NUM(4)
NUM(5)
NUM(6)
NUM(7)
NUM(8)
NUM(9)
NUM(10)
NUM(11)
NUM(12)
```

Het eerste teken van een array-variabele moet een letter zijn, de rest mag ook uit getallen bestaan. Evenals bij gewone variabelen mag u geen sleutelwoorden of interpunctiesymbolen opnemen in de namen van array-variabelen. Alleen de eerste twee tekens van een array-naam worden door de computer herkend en kleine letters worden niet onderscheiden van hoofdletters. Daarom zijn de volgende array-namen voor de computer identiek:

```
ab(10)
AB(10)
Abbacus(10)
```

Indien u strings in een array wilt plaatsen, moet u de daarvoor ontworpen string-array gebruiken. Het enige uiterlijke verschil met een numerieke array is dat een string-array moet eindigen met het '$'-symbool (evenals dat het geval is bij een string-variabele zelf).
Toegestane namen zijn bijvoorbeeld:

```
Game$(20)
Address$(10)
A1$(5)
```

De volgende namen zijn daarentegen niet toegestaan:

`TOWN$(12)`	*(bevat het TO-commando)*
`Question?$(6)`	*(interpunctietekens niet toegestaan)*
`name$(200)`	*(NAME is een gereserveerd woord)*

U kunt alle arrays die u in een programma nodig heeft in één opdracht opgeven aan het begin van uw programma:

```
DIM A$(12),B(13),C(20)
```

definieer nooit meer arrayruimte dan nodig is, omdat de computer na het lezen van de DIM-opdracht direct de benodigde ruimte in het geheugen reserveert, zodat alle ruimte die u niet gebruikt, wordt verspild.
U kunt geheugenruimte besparen door gebruik te maken van integere arrays in plaats van reële arrays, dit is NUM%(12) reserveert 13 integere variabelen, bestaande uit NUM%(0), NUM%(1), ..., NUM%(12).
Meerdimensionale arrays worden in het volgende hoofdstuk beschreven.

U kunt nu uw namen en nummers inlezen in het telefoonboek, met behulp van een lus:

```
10 REM telefoonboek versie 1
20 REM invoer arrays n$ en num$
30 CLS
40 INPUT "Aantal namen"; E
50 N=E-1
60 DIM N$(N), NUM(N)
70 FOR I=0 TO N
80 CLS
90 LOCATE ,10
100 INPUT "Naam      "; N$(I)
110 INPUT "Nummer    "; NUM$(I)
120 NEXT I
```

De variabele N wordt gebruikt in de DIM-opdracht in regel 40 om er zeker van te zijn dat hetzelfde aantal elementen wordt gereserveerd voor N$ en NUM als onderdelen in het telefoonboek.
N is één minder dan E, omdat deze het eerste element van ieder array-element (0) is en niet element (1).
Alle namen en telefoonnummers staan opgeslagen in arrays. Het volgende gedeelte van het programma laat het telefoonboek op het beeldscherm zien. Het programma wordt nu als volgt:

```
10 REM telefoonboek versie 2
20 REM invoer arrays n$ en num$
30 CLS
40 INPUT "Aantal namen"; E
50 N=E-1
60 DIM N$(N), NUM(N)
70 FOR I=0 TO N
80 CLS
90 LOCATE ,10
100 INPUT "Naam      "; N$(I)
110 INPUT "Nummer    "; NUM$(I)
120 NEXT I
130 REM toon telefoonboek
140 CLS
150 PRINT TAB(2) "NAAM" TAB(18) "NUMMER"
160 PRINTTAB(2) "**********************"
170 FOR I=0 TO N
180 LOCATE 2,4+I*2
190 PRINT N$(I)
200 LOCATE 18,4+I*2
210 PRINT NUM(I)
220 NEXT I
```

Als het programma klaar is, bent u al uw namen en nummers kwijt! Het is

mogelijk gegevens te bewaren aan het einde van een programma. U kunt dan met het programma de gegevens opslaan op tape.
Om dit te bereiken, kunt u gebruik maken van de READ- en DATA-commando's. Een DATA-opdracht kan worden gevolgd door een lange lijst van numerieke of alfabetische gegevens, gescheiden door komma's:

```
DATA Tom,4402073, Henry,2246607, Graham, 4421708, ...
```

De enige beperking in het aantal gegevenselementen is de totale hoeveelheid tekens in de regel; deze mag niet meer zijn dan 255.
U kunt zo veel DATA-opdrachten in uw programma opnemen als u wilt. U kunt ze overal waar u wilt plaatsen, omdat de computer DATA-opdrachten negeert, tot hij een READ-opdracht tegenkomt. Het is echter aan te raden om DATA-opdrachten altijd aan het einde van uw programma te plaatsen, omdat dit het wijzigen van de DATA-elementen vereenvoudigt (u hoeft niet het hele programma af te zoeken).
Laten we nu eens kijken naar het telefoonboekprogramma. In deze nieuwe versie van het programma zijn de gegevenselementen reeds opgenomen in de DATA-opdrachten, in plaats van invoer via INPUT. We weten het aantal elementen in het telefoonboek omdat we ze al in de DATA-opdrachten hebben geplaatst; we hoeven dus niet meer alles in te voeren via INPUT. Het aantal elementen wordt in de variabele E geplaatst, omdat u wellicht het aantal elementen later wilt kunnen wijzigen (dat bereikt u door alleen E aan te passen in regel 30). Het programma ziet er nu als volgt uit:

```
10 REM TELEFOONBOEK VERSIE 3
20 REM LEES GEGEVENS IN
30 E=4
40 N=E-1
50 DIM N$(N),NUM(N)
60 CLS
70 REM TOON GEGEVENS
80 PRINTTAB(2) "NAAM" TAB(18) "NUMMER"
90 FOR I=0 TO N
100 READ N$(I), NUM(I)
110 LOCATE 2,4+I*2
120 PRINT N$(I)
130 LOCATE 18, 4+I*2
140 PRINT NUM(I)
150 NEXT I
160 END
170 DATA BATMAN,774 0303, ROBIN, 4206734
180 DATA "NICO",023 344543, SCHEPER, 040 4444
```

Zodra de eerste READ-opdracht wordt uitgevoerd, leest de computer het eerste element uit de DATA-regel en plaatst deze in de variabele die in de READ-opdracht staat. In dit geval wordt SUPERMAN ingelezen in de variabele N$(0) van de array N$. De tweede keer dat een READ wordt uitgevoerd, wordt

het tweede DATA-element ingelezen. Zodra alle elementen in een DATA-regel zijn ingelezen, gaat de computer verder met de volgende regel en leest deze vervolgens in.
U zou kunnen denken dat het onnodig is om de DATA-elementen in te lezen met READ in een array. U zou dan gelijk kunnen hebben! Echter, zodra de gegevens in de array staan, kunnen we ze manipuleren, nog voordat ze worden getoond. Het volgende hoofdstuk geeft de uiteindelijke versie van het programma, waarin de gegevens alfabetisch zijn gesorteerd, voordat ze worden getoond.
U mag van meerdere READ-opdrachten gebruik maken om gegevens in te lezen vanuit een DATA-regel. De volgende READ-opdracht begint te lezen vanaf het eerste ongelezen DATA-element. Bijvoorbeeld:

```
10 READ A,B
20 READ A$,B$
30 PRINT A;B;A$;B$
40 DATA 10,20,"Dit is een regel.","HALLO"
```

levert op:

```
10 20Dit is een regel.HALLO
```

Wees voorzichtig met het gebruik van DATA-opdrachten. U moet er zeker van zijn dat de volgorde van de gegevens correspondeert met de volgorde van de READ-variabelen. Als u elementen in een numerieke variabele inleest, moet op dat moment een getal op de corresponderende positie binnen de DATA-regel staan.
U kunt geen variabelen opnemen in een DATA-regel, alleen maar strings en getallen. Strings in DATA-opdrachten hoeven niet tussen aanhalingstekens te worden geplaatst, tenzij ze komma's, puntkomma's of spaties bevatten.
Het kan gebeuren dat u de DATA-regel tweemaal nodig heeft. De eerste keer drukt u de gegevens bijvoorbeeld af, de tweede keer voert u er berekeningen mee uit. Om de computer terug te laten keren naar een reeds gelezen DATA-regel gebruikt u het RESTORE-commando.
RESTORE alleen laat de computer opnieuw lezen vanaf de eerste DATA-regel. RESTORE gevolgd door een regelnummer laat de computer vanaf de DATA-regel lezen die in de opgegeven regel zelf of daarna volgt. Overtuig u ervan dat u RESTORE met een correct nummer uitvoert, dit is een bestaand nummer in het programma. Start het volgende programma:

```
10 READ A,B
20 PRINT "A=";A,"B=";B
30 RESTORE
40 READ C,D
50 PRINT "C=";C,"D=";D
60 DATA 10,20
```

U krijgt te zien:

```
A=10          B=20
C=10          D=20
```

Hoofdstuk 13

Gegevensmanipulatie en sorteren

In het vorige programma werd een telefoonboekprogramma beschreven; het zou gemakkelijk zijn om het telefoonboek in alfabetische volgorde af te drukken. We hebben al een zeer beperkte vorm van lijstsorteren gezien in numerieke volgorde, in hoofdstuk 6. De nu volgende methode is veel flexibeler (zij het misschien niet de beste, maar wel één van de meest begrijpelijke methoden). Deze methode noemt men wel BUBBLE SORTING.
Laten we eens naar een lijst van getallen gaan kijken die moet worden gesorteerd:

```
7,4,5,10,2,8
```

We willen deze getallen naar aflopende waarde sorteren. Laten we eerst twee getallen vergelijken. Als het getal links kleiner is dan het getal rechts, dan verwisselen we ze, en vergelijken vervolgens het tweede met het derde getal. Stonden de eerste twee getallen reeds in de gewenste volgorde, dan waren we direct naar het tweede en derde getal gaan kijken.
Dus in ons voorbeeld vergelijken we eerst 7 met 4. Deze staan dus in de goede volgorde, zodat we meteen de volgende twee getallen, 4 en 5, vergelijken. 4 is minder dan 5, dus verwisselen we deze twee van plaats. We krijgen dan:

```
7,5,4,10,2,8
```

Nu vergelijken we het derde met het vierde getal, dit zijn 4 en 10. Ook nu weer verwisselen we de getallen:

```
7,5,10,4,2,8
```

Bij het vierde en vijfde getal wijzigt niets. Ten slotte vergelijken we het vijfde getal, 2, met het laatste getal, 8, en verwisselen deze van plaats zodat we uiteindelijk krijgen:

```
7,5,10,4,8,2
```

We zijn echter nog niet klaar! U ziet dat het kleinste getal achteraan is terechtgekomen. Laten we eens kijken wat een tweede doorloop oplevert:

VERGELIJK	VERWISSEL?	RESULTAAT
1ste/2de	nee	7,5,10,4,8,2
2de/3de	ja	7,10,5,4,8,2
3de/4de	nee	7,10,5,4,8,2
4de/5de	ja	7,10,5,8,4,2
5de/6de	nee	7,10,5,8,4,2

Aan het einde van de tweede doorloop bevinden de laagste twee getallen zich in de juiste volgorde aan het einde van de lijst. Het maximale aantal doorlopen om alle getallen in de juiste volgorde te krijgen, is daarom één minder dan het totale aantal elementen in de te sorteren lijst. In dit voorbeeld zou het aantal doorlopen dus vijf zijn. Soms kan een lijst al gesorteerd zijn, soms gedeeltelijk, soms helemaal. Het is daarom goed om na iedere doorloop te controleren of de lijst reeds is gesorteerd.
Laten we nu het telefoonboek eens sorteren op alfabetische volgorde. Het programma heeft drie onderdelen. Het eerste gedeelte leest de gegevens (READ, DATA) in, in een array, het tweede gedeelte sorteert de gegevens op alfabetische volgorde en het derde gedeelte toont de gesorteerde gegevens.

```
10 REM TELEFOONBOEK VERSIE 3
20 REM LEES GEGEVENS IN
30 E=4
40 N=E-1
50 DIM N$(N),NUM(N)
60 FOR I=0 TO N
70 READ N$(I), NUM(I)
80 NEXT I
90 REM SORTEREN
100 M=N-1
110 P=0
120 FOR C=0 TO M
130 IF N$(C) > N$(C+1) THEN SWAP N$(C),N$(C+1):SWAP NUM
(C),NUM(C+1):P=P+1
140 NEXT C
150 IF P<>0 GOTO 110
160 REM TOON GESORTEERDE GEGEVENS
170 CLS
180 PRINTTAB(2) "NAAM" TAB(18) "NUMMER"
190 PRINTTAB(2) "************************"
200 FOR I=0 TO N
210 LOCATE 2,4+I*2
220 PRINT N$(I)
230 LOCATE 18,4+I*2
240 PRINT NUM(I)
250 NEXT I
260 DATA KEES, 333333, JAN, 444444
270 DATA PIET, 111111, KLAAS, 222222
```

Het eerste en het laatste gedeelte zouden u bekend moeten zijn (zie hoofdstuk 12).
Regels 90 tot 150 bevatten de sorteerroutine. Aan het begin van de lus is de teller, P, gelijk aan nul. Tijdens iedere doorloop van de lus wordt een element in de array vergeleken met het naastliggende, volgende element, dus de eerste keer worden N$(0) en N$(1) met elkaar vergeleken. Staan de namen niet in alfabetische volgorde, dan worden de elementen van plaats verwisseld. Het bijbehorende telefoonnummer wordt meeverwisseld. P wordt nu op één gezet om aan te geven dat er een verwisseling heeft plaatsgevonden. De laatste keer dat de lus wordt doorlopen, wordt N$(2) met N$(3) vergeleken. De inhoud van deze elementen wordt verwisseld indien nodig, en de teller, P, verhoogd.
De uitvoering gaat dan verder bij regel 150. Hier controleert de computer de waarde van P. Als er elementen zijn verwisseld, is P groter dan 0. De computer veronderstelt dat de namen nog niet zijn gesorteerd en gaat terug naar regel 110. P wordt teruggezet op 0 en de sorteerlus wordt herhaald. Als bij regel 150 P=0 wordt gevonden, weet de computer dat geen elementen zijn verwisseld, zodat alle namen in alfabetische volgorde staan. De computer gaat dan verder naar de volgende regel en toont het telefoonboek. Als u het programma draait, krijgt u te zien:

```
NAAM                 NUMMER
************************
JAN                  444444
KEES                 333333
KLAAS                222222
PIET                 111111
```

Als u dezelfde arraynaam tweemaal in een programma wenst te gebruiken, doch met verschillende gegevens, moet u eerst de oude array uitwissen:

```
ERASE N$
```

Dit wist alle namen uit die waren opgeslagen in array N$. U kunt nu array N$ opnieuw gebruiken en er een dimensie aan toekennen met behulp van een DIM-opdracht.
Als u toevallig een variabele met dezelfde naam N$ heeft benoemd, wordt deze absoluut niet aangetast door ERASE. Probeer echter variabelen en arrays met dezelfde namen te vermijden, omdat dit nogal verwarrend kan zijn.
U kunt ook arrays met meer dimensies gebruiken, zelfs tot 255 dimensies aan toe! Voor dit moment beschouwen we een tweedimensionale array. Deze arrays zijn zeer handig voor het opslaan van tabellen:

DATUM	CREDIT	DEBET	BALANS
30.6.85	34.05		34.05
5.7.85		10.00	24.05
10.7.85	120.00		144.05

12.7.85		50.00	94.05
20.7.85	200.00		294.05

Deze vier kolommen met getallen kunt u opslaan in de tweedimensionale array, BANK(4,3). Deze array is groot genoeg om een tabel op te slaan die bestaat uit vijf rijen en vier kolommen, en zou als volgt kunnen worden voorgesteld:

BANK(0,0)	BANK(0,1)	BANK(0,2)	BANK(0,3)
BANK(1,0)	BANK(1,1)	BANK(1,2)	BANK(1,3)
BANK(2,0)	BANK(2,1)	BANK(2,2)	BANK(2,3)
BANK(3,0)	BANK(3,1)	BANK(3,2)	BANK(3,3)
BANK(4,0)	BANK(4,1)	BANK(4,2)	BANK(4,3)

Omdat dit een numerieke array betreft, moet de datum als een getal worden opgeslagen, bijvoorbeeld 300685. Iedere invoer voor deze tabel wordt opgeslagen in een apart element van de array. De datum 300685 kan worden opgeslagen in BANK(0,0), de uiteindelijke balans in BANK(4,3).
Het volgende programma toont u hoe u gegevens in een tweedimensionale array plaatst. De vier kolommen van de array worden benaderd via de waarden C=0 tot 3. Het aantal rijen hangt af van het aantal ingevoerde transacties, zodat, wanneer er T transacties worden gedaan, R de waarde van 0 tot T-1 kan aannemen. U voert de datum in van iedere transactie en het bedrag aan DEBET of CREDIT, waarna de BALANS wordt uitgerekend voor iedere transactie en het resultaat in de vierde kolom van de array wordt geplaatst. Het totaal aan CREDIT, DEBET en BALANS wordt opgeslagen in de laatste rij van de array.
De titelregel van de tabel wordt bewaard in een aparte string-array. Deze hoeft geen dimensie te hebben, omdat het een ééndimensionale array betreft, die minder dan elf elementen bevat. Alle multi-dimensionale arrays moeten voor het gebruik van een dimensie worden voorzien.

```
10 REM BEREKEN BANKSALDO
20 REM LEES TABELKOPREGELS IN
30 CLS
40 FOR C=0 TO 3
45 READ KOP$(C)
50 NEXT C
60 REM INVOER IN 2D ARRAY
70 INPUT "AANTAL TRANSAKTIES";T
80 DIM BANK(T,3)
90 N=T-1
100 CLS
110 FOR C=0 TO 2
120 PRINTTAB(10*C) KOP$(C);
130 NEXT C
140 FOR R=0 TO N
150 FOR C=0 TO 2
160 LOCATE 10*C,R+3
170 INPUT BANK(R,C)
```

```
180 NEXT C
190 NEXT R
200 REM BEREKEN BALANS NA ELKE TRANSAKIE
210 BANK(0,3)=BANK(0,1)-BANK(0,2)
220 FOR R=1 TO N
230 X=BANK(R,1)-BANK(R,2)
240 BANK(R,3)=BANK(R-1,3)+X
250 NEXT R
260 REM BEREKEN TOTAAL BALANS
270 FOR R=0 TO N
280 BANK(T,1)=BANK(R,1)+BANK(T,1)
290 BANK(T,2)=BANK(R,2)+BANK(T,2)
300 NEXT R
310 BANK(T,3)=BANK(T,1)-BANK(T,2)
320 BANK(T,0)=BANK(N,0)
330 REM TOON BANKBALANS
340 SCREEN 0
350 WIDTH 40
360 FOR C=0 TO 3
370 PRINTTAB(10*C) KOP$(C);
380 NEXT C
390 FOR R=0 TO N
400 FOR C=0 TO 3
410 LOCATE 10*C,R+2
420 PRINT BANK(R,C)
430 NEXT C
440 NEXT R
450 PRINT "________________________________________"
460 FOR C=0 TO 3
470 PRINTTAB(10*C) BANK (T,C);
480 NEXT C
490 END
500 DATA DATUM,KREDIT,DEBET,BALANS
```

Als u DEBET of CREDIT invoert groter dan 99999.99, dan wordt de tabel onoverzichtelijk, omdat maar tien posities zijn gereserveerd voor iedere invoer.

Oefening

1. Verander het programma door een kleine ingreep, opdat u de datum kunt invoeren in het formaat 23.6.85 enz. U heeft hier een andere string-array voor nodig; DATA$(T) is hier geschikt voor.

Hoofdstuk 14

Het manipuleren van strings

Er bestaan heel wat verschillende manieren om strings te manipuleren. Tot nu toe heeft u alleen strings bij elkaar gevoegd:

```
PRINT "GOEDEN"+"DAG"
```

wordt afgedrukt als:

```
GOEDENDAG
```

Met behulp van INSTR kunt u binnen een string zoeken naar een bepaalde string. In een avonturenprogramma zou het kunnen voorkomen dat u de speler vraagt welke richting hij of zij op wil gaan. Het programma springt naar verschillende plaatsen, afhankelijk van wat wordt ingetikt, bijv. noord, west, zuid of oost. Het volgende programmagedeelte geeft aan hoe dit zou kunnen worden gedaan:

```
100 INPUT "Waar gaat u nu heen ";B$
110 IF B$="noord" GOTO 200
120 IF B$="oost" GOTO 300
130 IF B$="zuid" GOTO 400
140 IF B$="west" GOTO 500
150 PRINT "Probeer het nog eens!": GOTO 100
```

De speler zou echter een complete zin kunnen hebben ingetikt, als antwoord op de vraag in regel 100, bijv. 'Ik ga nu oostwaarts'.
Zoals het programma nu is, zal geen van de sprongopdrachten worden uitgevoerd en zal opnieuw worden gevraagd welke richting de speler op wil gaan. Dit blijft zo doorgaan tot het enkele woord 'oost' wordt ingevoerd, hetgeen het spel aanzienlijk doet vertragen. Het zou veel beter zijn wanneer de computer de hele zin controleerde op de aanwezigheid van een bepaalde string. Dit is in feite wat INSTR doet. Het volgende voorbeeld laat zien hoe:

```
100 INPUT "Waar gaat u nu heen";B$
110 IF INSTR(B$,"noord")<>0 GOTO 200
120 IF INSTR(B$,"west")<>0 GOTO 300
130 IF INSTR(B$,"zuid")<>0 GOTO 400
```

```
140 IF INSTR(B$,"oost")<>0 GOTO 500
150 PRINT "Probeer het nog eens!": GOTO 100
```

In regel 110 doorzoekt INSTR de hoofdstring, B$, en controleert deze op de aanwezigheid van de kleinere string 'noord'. Als deze wordt gevonden, wordt de waarde teruggegeven die correspondeert met de positie van de eerste letter van 'noord' in de hoofdstring.
Als 'oost' niet wordt gevonden, geeft INSTR een nul terug.
Dus als B$ is:

```
"Ik ga nu noordwaarts"
```

geeft INSTR(B$,"noord") de waarde 10 terug, overeenkomend met de 'n' van 'noord' die op de tiende tekenpositie staat. Om dit te controleren op juistheid, kunt u het volgende korte programma intikken:

```
10 A=INSTR("Ik ga nu noordwaarts","noord")
20 PRINT A
```

Het getal 10 zou nu moeten worden afgedrukt. INSTR geeft altijd een getal terug en moet daarom dus altijd met een numerieke variabele worden vergeleken.
Let erop dat u de strings in een INSTR-opdracht tussen aanhalingstekens plaatst. String-variabelen zijn reeds voorzien van aanhalingstekens, dus deze kunnen ongewijzigd worden opgenomen in de INSTR-opdracht.
Voor het doorzoeken van een string kunt u vanaf een willekeurige positie starten:

```
INSTR(10,A$,B$)
```

Dit laat het zoeken naar B$ binnen A$ beginnen bij of vanaf de 10e tekenpositie in de string.
Dit INSTR-commando zou heel goed kunnen worden toegepast in het bekende 'galgje'-spelprogramma. Het volgende programma doorzoekt het woord:

```
CONSTANTINOPEL
```

voor de letter 'N':

```
10 A$="CONSTANTINOPEL"
20 B$="N"
30 C=0
40 C=INSTR(C+1,A$,B$)
50 IF C=0 THEN END
60 PRINT "ER STAAT EEN ";B$" OP TEKENPOSITIE ";C
70 GOTO 40
```

In dit programma geeft C+1 aan waar met zoeken dient te worden begonnen. De eerste keer dat INSTR een 'N' vindt, staat C op 3, zodat de

daaropvolgende zoekactie vanaf positie 4 begint. Nadat de derde 'N' is gevonden, wordt C=10 en ten slotte start de vierde en tevens laatste zoekactie vanaf positie 11. Er zijn nu niet meer 'N's, dus C=0, waardoor het programma stopt door een END-opdracht in regel 50.
U krijgt altijd een nul terug wanneer u een string opzoekt binnen een kleinere string:

```
INSTR("dag","goedendag")
```

levert een nul op.
Nu volgen nog drie vrijwel gelijke functies:

```
RIGHT$, LEFT$ en MID$
```

Als eerste RIGHT$. Hier volgt een voorbeeld van het gebruik ervan:

```
10 A$=RIGHT$("Het is vandaag verschrikkelijk weer",4)
20 PRINT A$
```

levert op:

```
weer
```

De vier rechtse tekens zijn in de string-variabele A$ geplaatst.
U zult reeds vermoeden wat LEFT$ doet! Hier volgt toch nog een voorbeeld:

```
10 A$=LEFT$("Het is vandaag verschrikkelijk weer",14)
20 PRINT A$
```

levert op:

```
Het is vandaag
```

Ten slotte MID$:

```
10 A$=MID$("Het is vandaag verschrikkelijk weer",15,16)
```

levert op:

```
verschrikkelijk
```

Het eerste getal in de MID$-opdracht staat voor de eerste tekenpositie die moet worden teruggegeven. Het tweede getal geeft het aantal terug te geven tekens aan. In het bovenstaande voorbeeld worden dus de eerste 16 tekens, vanaf positie 15 (verschrikkelijk + spatie), in de string-variabele A$ geplaatst.
Het getal in deze functies mag liggen tussen 1 en 255. U kunt trouwens toch geen regel invoeren die meer dan 255 tekens bevat, vandaar.
Als het eerste getal in MID$ groter is dan het aantal tekens in de string,

wordt een lege string geretourneerd. Opnieuw moet u er aan denken aanhalingstekens mee te geven aan de strings, en niet proberen één van deze commando's te vergelijken met een numerieke variabele; u dient stringvariabelen te gebruiken omdat strings worden teruggegeven door deze functies.

Oefeningen

1. Door gebruik te maken van INSTR, moet u het resultaat vinden van:

   a. het zoeken naar een lege string of nulstring, dit is " ";
   b. het zoeken naar een string binnen een lege string;
   c. het zoeken naar een lege string binnen een lege string.

2. Druk de woorden 'Is daar iemand' separaat af. U zou INSTR kunnen gebruiken om de positie van ieder woord te bepalen, en vervolgens het gebruik van MID$ herhalen, of gebruik maken van RIGHT$, LEFT$ en MID$.

3. Wijzig het laatste programma opdat een willekeurig ingevoerde string per woord wordt afgedrukt.

Hoofdstuk 15

Functies

U heeft reeds met enkele functies kennis gemaakt, bijvoorbeeld:

INT	hoofdstuk 4
INSTR	hoofdstuk 14
LEFT$	hoofdstuk 14
MID$	hoofdstuk 14

In het algemeen gedragen deze functies zich op dezelfde wijze. U heeft een object, dit is het argument, waarop de functie wordt losgelaten, zodat deze een resultaat produceert. Als een functie is gedefinieerd, genereert hij altijd zijn resultaten op dezelfde wijze indien een geschikt argument wordt meegegeven:

```
INT(X)
```

X stelt het argument voor. Het argument van een functie staat altijd tussen haakjes. Het argument moet een getal, een numerieke variabele of een numerieke uitdrukking zijn. Zolang X hieraan voldoet, geeft INT altijd een integere waarde terug, door X af te ronden naar het direct eronder gelegen gehele getal:

INT(7.9)	geeft 7
INT(-2.3)	geeft -3

Als u in de war raakt met negatieve getallen, bekijk dan de volgende regel eens:

... -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 ...

De getallen staan geordend naar toenemende waarde, van links naar rechts. Hieruit kunt u afleiden dat -3 in feite het dichtstbijgelegen gehele getal is van -2.3.

Er bestaat nog een soortgelijke functie als INT, nl. FIX. FIX reageert ook op een reëel getal als argument door een integer getal terug te geven. In het geval van positieve getallen, geeft FIX de dichtstbijgelegen lagere integere waarde evenals bij INT, maar voor negatieve getallen geeft FIX de

dichtstbijgelegen hogere integere waarde, in tegenstelling tot INT:

FIX(5.5)	geeft 5
FIX(-2.2)	geeft -2

Nu volgen nog twee functies met numerieke argumenten, ABS en SGN. SGN geeft het teken van een getal:

Als het argument een negatief getal is, geeft de functie -1 terug.
Als het argument 0 is, geeft de functie 0 terug.
Als het argument een positief getal is, geeft de functie +1 terug.

```
10 A=SGN(-402.2)
20 PRINT "SGN(-402.2)=";A
```

geeft:

```
SGN(-402.2)=-1
```

ABS daarentegen, wordt gebruikt om negatieve getallen naar positieve waarden om te zetten. Positieve waarden worden daarom door deze functie niet veranderd:

```
10 A=ABS(-402.2)
20 PRINT "ABS(-402.2)=";A
```

geeft:

```
ABS(-402.2)= 402.2
```

Het argument kan bij beide functies zowel een uitdrukking als een variabele zijn:

SGN(4+3*-2)	geeft -2
ABS(procent)	zet de waarde opgeslagen in procent om naar een positief getal.

Hieronder treft u een kort programma aan dat een mogelijke toepassing van ABS illustreert:

```
10 REM DIT PROG BEREKENT UW LEEFTIJD
20 INPUT"VANDAAG IS HET: DAG, MAAND, JAAR";D,M,J
30 INPUT"UW GEBOORTEDATUM IS: DAG, MAAND, JAAR";GD,GM,G
J
40 DA=ABS(D-GD)
50 MA=ABS(M-GM)
60 JA=ABS(J-GJ)
70 PRINT "U BENT";JA;"JAREN,";MA;"MAANDEN,";DA;"DAGEN."
```

Nu volgen twee functies die zowel strings als getallen behandelen, VAL en STR$.
Als een string een getal bevat, dan geeft VAL de numerieke voorstelling ervan terug, dit is het verwijderen van de aanhalingstekens en spaties binnen de string. Bijvoorbeeld:

```
A=VAL("   273")
PRINT "A=";A
```

geeft:

```
A= 273
```

Indien er meer dan één getal in een string staat, geeft VAL alleen het eerste getal terug dat hij tegenkomt. Om deze reden evalueert VAL geen uitdrukkingen binnenin strings:

```
PRINT VAL("3+4=7")
```

geeft:

```
3
```

Vergeet niet dat, ook al is het argument van VAL een string, het door VAL teruggegeven resultaat een getal is en daarom in een numerieke variabele moet worden opgeslagen:

```
A$=VAL(7)                IS FOUT!
```

VAL kan niet overweg met getallen die tussen letters in staan:

```
10 A=VAL("Ik ben vandaag 20 jaar geworden.")
20 PRINT A
```

geeft:

```
0
```

De tegenovergestelde functie van VAL wordt uitgevoerd door STR$; deze zet een numeriek argument om naar een string:

```
A$="Ik ben vandaag "+STR$(60)+" jaar geworden."
PRINT A$
```

geeft:

```
Ik ben vandaag 60 jaar geworden.
```

Vergeet hier niet, dat ook al is het argument binnen de functie een getal,

het resultaat een string is en derhalve alleen in een string-variabele kan worden opgeslagen.
In hoofdstuk 13 gebruikten we een numerieke array, BANK, om datum-, credit-, en balansgegevens in op te slaan. Helaas moest toen de datum als een getal, dit is 240685, worden ingevoerd, in plaats van bijvoorbeeld 24/6/85. Invoer volgens de tweede vorm zou alleen kunnen door de datum in een aparte string-array op te nemen, zodat het programma uiteindelijk een numerieke array voor de transacties en twee string-arrays voor de titelregels en data had. Door nu STR$ te gebruiken, kunnen alle gegevens in een lange string-array worden opgeslagen. Om dit te bereiken, zijn de volgende wijzigingen nodig:

```
170 INPUT BANK
175 BANK$(R,C)=STR$(BANK)
```

Om de nieuwe balans op te maken na iedere transactie, moet u eerst de strings evalueren in de kolommen credit en debet via VAL, en vervolgens het resultaat weer omzetten tot een string:

```
210 BANK$(0,3)=STR$(VAL(BANK$(0,1))-VAL(BANK$(0,2)))
```

Misschien wilt u nu eerst even teruggaan om het BANK-BALANS-programma aan te passen.
De laatste functie die betrekking heeft op een string, is LEN. LEN geeft u de lengte terug van de string in het argument:

```
10 A$="CONSTANTINOPEL"
20 A=LEN(A$)
30 PRINT "Het woord ";A$;" is ";A;" tekens lang."
```

Opnieuw dient u er op te letten dat ook al wordt een string als argument opgegeven, het resultaat een getal is, zodat deze alleen mag worden gelijkgesteld aan een numerieke variabele.
Ten slotte nog een heel andere functie, namelijk RND. RND genereert een 14-tallig willekeurig getal, liggend tussen 0 en 1. In werkelijkheid zijn deze getallen niet echt willekeurig, omdat zij in een vaste volgorde staan. Deze serie getallen is echter dermate lang dat u de geproduceerde getallen als echte willekeurige getallen mag beschouwen.
U kunt drie verschillende soorten argumenten meegeven aan RND:

Wanneer het argument positief is, is het door RND gegenereerde getal de eerstvolgende in de volgorde. Het punt waar de volgorde wordt gestart, is altijd hetzelfde en wordt door de computer teruggezet aan het einde van een programma. Tik eens in:

```
10 PRINT RND(1)
20 PRINT RND(1)
30 PRINT RND(1)
```

U krijgt telkens de volledige reeks resultaten, per keer dat het programma wordt gestart:

```
.59521943994623
.10658628050158
.76577651772823
```

Het wijzigen van het argument in een ander positief getal verandert hier niets aan.

Als het argument 0 is, is het resultaat gelijk aan het vorige. Bijvoorbeeld:

```
10 X=RND(1)+RND(1)
20 PRINT "Het tweede willekeurige getal was ";RND(0)
30 PRINT "Het eerste willekeurige getal was ";X-RND(0)
```

Als het argument negatief is, wordt het specifieke deel van de reeks van waaruit de getallen worden gegenereerd, gezet door het negatieve argument:

```
10 A=RND(-3)
20 B=RND(1)
30 PRINT A,B
```

levert de volgende reeks:

```
.84389820420821             .29624868166920
```

steeds als het programma wordt gedraaid. Vervangt u nu -3 door -4 dan krijgt u een heel andere reeks. Het wijzigen van het positieve getal heeft ook hier geen effect.

Een willekeurig getal tussen 0 en 1 is zelden bruikbaar. Het is veel waarschijnlijker dat u bijvoorbeeld een getal tussen 1 en 6 wilt hebben. Het volgende programma simuleert de worp van een dobbelsteen:

```
10 REM dobbelsteen-programma
20 A=INT(6*RND(1)+1)
30 PRINT "U heeft een ";A;" geworpen."
```

Echter, u krijgt steeds opnieuw hetzelfde resultaat. Wilt u steeds een ander willekeurig getal, voeg dan de volgende regel toe:

```
RDRST=RND(-TIME)
```

aan het begin van uw programma.
De TIME-functie geeft de waarde van de interne klok van de computer. Na het inschakelen van de computer staat de klok op 0, maar wordt iedere 1/50 van

een seconde verhoogd, dus TIME is gelijk aan 50 na het verstrijken van een volle seconde. Door TIME te gebruiken als argument van RND, wordt bij iedere uitvoering van RND een ander deel van de reeks willekeurige getallen gelezen.
Ziet u een toepassing van dit en de laatst beschreven functie in het spel galgje? Het spel krijgt de volgende structuur:

1. Lees een willekeurig woord vanuit een DATA-regel (deze regels kunnen aan het einde van uw programma worden geplaatst, na END). Voor het willekeurig lezen van een woord, kunt u gebruik maken van RESTORE gevolgd door een RND met als argument een regelnummer. Bedenk eens, hoe u toegang tot meer dan één woord in een DATA-regel kunt krijgen.
2. Als u een woord heeft gelezen, laat dan de speler weten wat de lengte van het woord is, met behulp van LEN.
3. Vraag de speler een letter te raden.
4. Met behulp van INSTR controleert u de aanwezigheid van de gekozen letter in het woord. (Zie hoofdstuk 14 als u vergeten bent hoe u dit kunt doen.)
5. Tel het aantal keren dat de speler raadt.
6. Als het woord na tien keer raden nog niet is gevonden, beëindigt u het programma. De computer heeft dan gewonnen.
7. Vraag de speler of hij/zij nog eens wenst te spelen, met behulp van INKEY$.

Oefening

1. Schrijf het galgje-programma!

Hoofdstuk 16

Het zelf definiëren van functies

Tot nu toe hebben we alleen de functies besproken die de computer biedt, vandaar dat we nu onze eigen functies gaan definiëren. Dit kan worden gedaan met behulp van DEF FN.
Zodra u een functie heeft gedefinieerd en deze een naam heeft gegeven, kunt u hem overal in het programma gebruiken.
Laten we als voorbeeld een wiskundige functie nemen. Stel u wilt het kwadraat weten van een getal, dit is het verheffen tot de macht twee, wat hetzelfde is als vermenigvuldigen met zichzelf.
Laten we deze functie KWADR noemen en als volgt definiëren:

```
DEF FNKWADR(X)=X^2
```

Let er op dat de naam van de functie direct na het commando DEF FN komt. X is wat men noemt een 'domme parameter'. U gebruikt hem alleen om aan te geven wat de functie doet. Tijdens het gebruik van de functie moet u een echte parameter meegeven om daarmee de domme parameter te vervangen. In het voorbeeld kan deze parameter zowel een numerieke variabele als een getal zijn. Bij het definiëren van een functie moet de domme parameter tussen haakjes worden geplaatst, achter de functienaam.
Laten we de functie eens gebruiken om het kwadraat van 13 te berekenen, om vervolgens het resultaat in variabele R te plaatsen. Om dit te bereiken, gebruiken we het commando FN, gevolgd door de functienaam en de echte parameter tussen haakjes:

```
R=FNKWADR(13)
```

Na het lezen hiervan kijkt de computer naar de functie KWADR. Na het vinden van de functie, vervangt de computer alle X-en door 13, berekent het resultaat en slaat dit op in de variabele R.
X wordt ook wel een plaatselijke variabele genoemd. Dit betekent dat hij alleen plaatselijk wordt herkend, dit is binnen de functie:

```
10 X=4
20 DEF FNKWADR(X)=X^2
30 R=FNKWADR(12)
40 PRINT "R=";R;"X=";X
```

geeft:

```
R=144 X=4
```

U kunt meer dan één parameter meegeven in een functie. Alle te gebruiken parameters dienen te worden vermeld, gescheiden door komma's. Bij het aanroepen van de functie moet u dezelfde hoeveelheid echte parameters meegeven als dummy-parameters die in de functie staan. U dient ook de juiste soort parameter mee te geven, dit is een getal of numerieke variabele voor een numerieke domme parameter.
Het volgende programma verheft een willekeurig getal tot de macht van een ander getal, waarbij beide getallen worden aangeboden aan de functie:

```
10 DEF FNMACHT(X,Y)=X^Y
20 INPUT "Getal";A
30 INPUT "Exponent";B
40 R=FNMACHT(A,B)
50 PRINT A;" ^ ";B;"=";R
```

De definitie van een functie moet in een programmaregel passen.
Nu een voorbeeld met strings. De volgende functie hakt het eerste teken van een string af:

```
10 DEF FNST$(A$)=RIGHT$(A$,LEN(A$)-1)
20 INPUT "STRING";S$
30 PRINT S$,FNST(S$)
```

Omdat deze functie een string teruggeeft, moet de functienaam eindigen op '$'. Iedere juiste naam voor een variabele is acceptabel als functienaam.

Oefening

1. Definieer een functie die de laatste vier tekens van een string afhakt.

Hoofdstuk 17

Gestructureerd programmeren

Het zal u vaak overkomen dat u in een programma dezelfde opdracht meerdere keren wilt uitvoeren, doch op verschillende plaatsen in het programma. Wanneer het inderdaad nodig blijkt een opdracht of procedure te herhalen, is het aan te bevelen de desbetreffende programmaregels in een zogenaamde 'subroutine' te plaatsen. Steeds als u de opdracht nodig heeft, springt u even naar de subroutine, met behulp van de GOSUB-opdracht. Zodra de subroutine zijn werk heeft gedaan, keert u terug, via RETURN, naar het hoofdprogramma. Een subroutine mag zoveel programmaregels bevatten als nodig blijkt te zijn. In het volgende programma bevindt de subroutine zich tussen de regels 100-130:

```
10 CLS
20 PRINT"VINDT U MIJ AARDIG?"
30 GOSUB 110
40 PRINT"DENKT U DAT IK SLIM BEN?"
50 GOSUB 110
60 PRINT "VINDT U MIJ KNAP?"
70 GOSUB 110
80 PRINT "DAT ZEGT U MAAR!"
90 END
100 REM SUBROUTINE
110 INPUT "JA OF NEE ";JN$
120 IF JN$="NEE" THEN PRINT " OK. DAN GA IK MAAR WEER."
:END
130 RETURN
```

Let op de RETURN-opdracht aan het eind van de subroutine. Dit is zeer belangrijk, omdat deze de computer aangeeft dat teruggesprongen dient te worden naar het hoofdprogramma. De uitvoering wordt dan voortgezet vanaf de regel die volgt op de regel met de GOSUB-opdracht. Het is aan te bevelen om een REM-opdracht vooraan in de subroutine te plaatsen, om deze zo te kunnen onderscheiden van het hoofdprogramma. Gebruik het regelnummer van de REM-opdracht niet als aanroep van de subroutine, omdat u later misschien besluit alle REM-opdrachten te verwijderen, ingeval u met een geheugentekort te kampen krijgt.
Wees er zeker van dat de computer de subroutine niet per ongeluk

binnenloopt, door na het programma een END-opdracht te plaatsen, vlak voor het begin van de subroutines.
Een andere toepassing van GOSUB is bij de IF...THEN...ELSE-opdracht. Deze opdracht neigt altijd lang en gecompliceerd te worden; de simpelste manier om deze te vereenvoudigen is door gebruik te maken van een subroutine.
Het volgende korte programma vraagt om twee verschillende positieve getallen. De IF...THEN...ELSE-opdracht wordt gebruikt om de getallen te controleren. Indien zij verschillen en positief zijn, worden het produkt en de som ervan afgedrukt. Anders wordt opnieuw om invoer gevraagd:

```
10 INPUT "Twee positieve, verschillende
getallen";A,B
20 IF A=B OR A<0 OR B<0 THEN PRINT 'Probeer
het nog eens!" ELSE PRINT "A=";A,"B=";B:
PRINT "A+B=";A+B:PRINT "AxB=";A*B
30 GOTO 10
```

Nu met behulp van een subroutine:

```
10 INPUT "TWEE POSITIEVE, VERSCHILLENDE GETALLEN";A,B
20 IF A=B OR A<0 OR B<0 THEN PRINT "PROBEER HET NOG EEN
S !" ELSE GOSUB 40
30 GOTO 10
40 REM SUBROUTINE
50 PRINT "A=";A,"B=";B
60 PRINT "A+B=";A+B
70 PRINT "AxB=";A*B
80 RETURN
```

Het programma wordt hierdoor iets langer, maar veel eenvoudiger te lezen.

Hoofdstuk 18

Programmasprongen

Soms zal het nodig zijn dat er een hoofdprogramma is, of menuprogramma, dat een aantal mogelijkheden biedt. Zo kunnen de eerste regels van het menuprogramma er voor zorgen dat het volgende op het beeldscherm verschijnt:

```
******** MENU ********

1. Toon telefoonboek
2. Voeg een element toe
3. Verwijder een element
4. Beëindig het programma

**********************
```

Het daaropvolgende programmagedeelte zou er als volgt kunnen uitzien:

```
100 A$=INKEY$
120 IF A$="1" THEN GOSUB 1000
130 IF A$="2" THEN GOSUB 2000
140 IF A$="3" THEN GOSUB 3000
150 IF A$="4" THEN GOSUB 4000
160 GOTO 100
```

Regel 160 is opgenomen om een toetsaanslag anders dan 1, 2, 3 of 4 te besturen.
Er bestaat een kortere manier om dit te schrijven, door het aanwenden van ON...GOSUB. Door gebruik te maken van deze opdracht kunt u naar elke gewenste hoeveelheid subroutines springen. In bovenstaand voorbeeld willen we naar één van de vier subroutines springen; daarom geven we de reeks regelnummers mee achter de GOSUB-opdracht:

```
ON...GOSUB 1000,2000,3000,4000
```

Nu moeten we de computer nog laten weten naar welke subroutine dient te worden gesprongen. Dit doen we door een getal of uitdrukking te plaatsen ná ON en vóór GOSUB. Als de uitdrukking een 1 oplevert, springt de computer

naar de eerste subroutine na de GOSUB; is het resultaat een 2, dan wordt de tweede subroutine aangeroepen enz.
We kunnen nu daarom de regels 100-160 vervangen door:

```
100 A$=INKEY$
110 A=VAL(A$)
120 ON A GOSUB 1000,2000,3000,4000
130 GOTO 100
```

Let er op dat de string-variabele A$ is veranderd in de numerieke variabele A, voordat deze wordt gebruikt in de ON...GOSUB-opdracht; anders zou er een fout zijn opgetreden, omdat alleen numerieke variabelen mogen volgen op het woord ON.
Na het beëindigen van de subroutine keert de computer terug naar regel 130.
Als de variabele een reëel getal had bevat, bijv. 2.9, dan had de computer de dichtstbijgelegen lagere integere waarde genomen en zou dus naar de tweede subroutine zijn gesprongen. De uitdrukking moet resulteren in een waarde, liggend tussen 1 en het aantal meegegeven regelnummers.
ON...GOTO werkt hetzelfde als ON...GOSUB. In dit geval springt de computer naar het regelnummer dat correspondeert met het nummer dat volgt op de ON-opdracht. De uitvoering van het programma gaat door vanaf dit regelnummer:

```
10 INPUT "1,2 of 3";A
20 ON A GOTO 370,420,10
```

Dus als:

A=1	is de volgende uit te voeren regel, regel 370
A=2	is de volgende uit te voeren regel, regel 420
A=3	is de volgende uit te voeren regel, regel 10

Let er op dat de regelnummers niet op volgorde hoeven te staan in de lijst.

Hoofdstuk 19

Wiskundige functies

Mocht u niet over wiskundige kennis beschikken en als derhalve blijkt dat dit hoofdstuk u zwaar valt, aarzel dan niet naar het volgende hoofdstuk door te gaan.
Dit hoofdstuk behandelt de volgende wiskundige functies:

```
^, SQR
TAN, SIN, COS en ATN
EXP en LOG
```

U weet al hoe u machten verheft met behulp van het symbool:

`4^2=16` (of zoals gewoonlijk geschreven $4^2=16$)

In het algemeen komt het er op neer dat het verheffen van een getal n tot de macht p, gelijk is aan het p keer vermenigvuldigen van n met zichzelf. In bovenstaand voorbeeld is n=4, p=2. Dus, 4 ^ 2 is gelijk aan 4*4.
De vierkantswortelfunctie, SQR, geeft de vierkantswortel van een getal. Dit is het tegenovergestelde van het verheffen van een getal tot de macht twee of het vermenigvuldigen van een getal met zichzelf.
Het kwadraat van 5, dit is 5 ^ 2, levert 25 op. Om terug te gaan van 25 naar 5, neemt u de vierkantswortel van dat getal (SQR):

```
10 A=SQR(25)
20 PRINT "De vierkantswortel uit 25 is ";A
```

hetgeen oplevert:

```
De vierkantswortel uit 25 is 5
```

Het argument van de functie dient een positief getal te zijn of een uitdrukking die resulteert in een positief getal. (De computer kan niet overweg met imaginaire getallen!)
Hier volgen nog enkele punten om over na te denken.
Wat gebeurt er als u een getal tot een negatieve macht verheft? U kunt dit

heel gemakkelijk uitproberen door in te tikken:

```
PRINT 4 ^ -1
```

U krijgt dan:

```
.25
```

wat 1/4 is, dus in het algemeen:

```
n ^ -p = 1/n^p
```

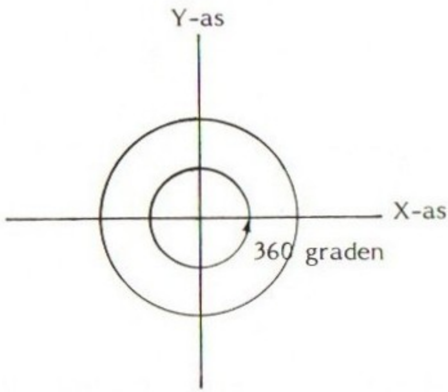
Laten we eens kijken wat er gebeurt als u een getal verheft tot de macht van een decimaal getal of breuk. Tik in:

```
PRINT 4 ^ 0.5
```

geeft als antwoord 2.
4 ^ 0.5 is gelijk aan 4½, dus in het algemeen:

`n^1/p` levert de p-de wortel uit n.

In het voorbeeld is n=4, p=2, zodat u de tweedemachtswortel, of beter bekend als vierkantswortel uit 4 krijgt, die 2 is. Dus het verheffen van een getal tot de macht ½, is gelijk aan de vierkantswortel van het getal.
SIN, COS, TAN en ATN zijn trigonometrische functies. De argumenten binnen deze functies moeten hoeken zijn. U bent wellicht gewend aan graden als u een hoek wilt uitdrukken:



Een cirkel beschrijft 360° om zijn centrum. De hoek van de cirkel wordt gemeten vanaf de X-as, die de cirkel in tweeën deelt, door een lijn op de negen-uur-positie en de drie-uur-positie, in de tegenovergestelde richting van de wijzers van de klok. De Y-as deelt de cirkel in tweeën van de

twaalf-uur-positie tot de zes-uur-positie. De hoek tussen de twee assen is daarom ook 90°.
De computer meet de hoeken echter heel anders. Om te begrijpen hoe beschouwen we een cirkel met een straal van één eenheid. Het soort eenheid is niet van belang; deze kan 1 mm, 1 m, 1 km - alles zijn.
De omtrek van de cirkel wordt gegeven door de formule:

C=2*PI*R	C=Omtrek PI is een speciaal wiskundig getal, dat begint met 3.1415926... PI is Grieks voor 'p' en wordt uitgesproken als 'PIE' R is de straal van de cirkel

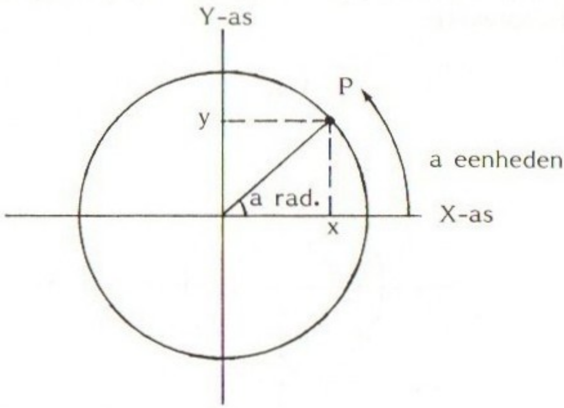
Daarom geldt voor onze cirkel, met R=1:

C=2*PI

Door hier nog eens op een andere manier naar te kijken, kunnen we zien dat de omtrek van onze cirkel een hoek van 2*PI radialen onderspant vanaf het centrum van de cirkel. Dus:

2*PI radialen is gelijk aan 360°.

De hoek tussen de X-as en Y-as is daarom in radialen uitgedrukt, PI/2. Als we nu een punt langs de omtrek van onze cirkel laten bewegen, kunnen we met behulp van de functies COS en SIN bepalen waar dit punt ligt:



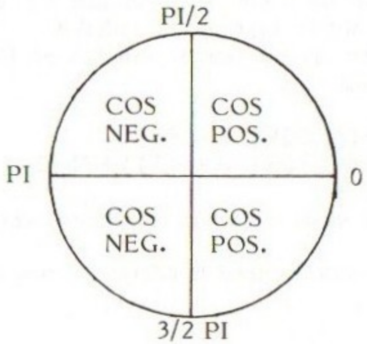
Op een bepaald moment bevindt het punt zich op positie P, waar het een hoek heeft doorlopen ten opzichte van de X-as van a radialen of als u dat liever is, a eenheden van de omtrek.
Als we nu verticale en horizontale stippellijnen trekken vanuit P, zodat we de X-as en Y-as snijden, vinden we de positie van de snijpunten op de X-as en de Y-as, door achtereenvolgens gebruik te maken van COS en SIN:

Y=COS(a) COS staat voor de de cosinus van hoek a
Y=SIN(a) SIN staat voor de sinus van hoek a

Als we langs de X-as meten, beschouwen we x als positief, als we ons rechts van de Y-as bevinden, en x is negatief links van de Y-as. Daarom kunnen we stellen:

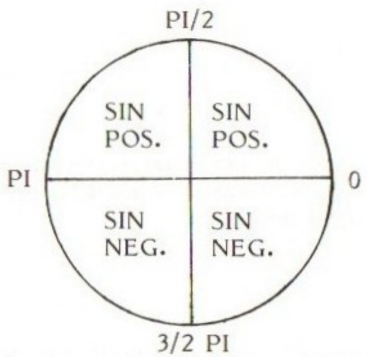
De COS van een hoek die ligt tussen 0 en PI/2 is positief.
De COS van een hoek die ligt tussen PI/2 en 3*PI/2 is negatief.
De COS van een hoek die ligt tussen 3*PI/2 en 2*PI is positief.

Dus voor COS geldt:



Als we langs de Y-as meten, dan is y positief als we boven de X-as zijn, en negatief als we er onder zitten. Hiervoor kunnen we dus stellen:

De SIN van een hoek die ligt tussen 0 en PI is positief.
De SIN van een hoek die ligt tussen PI en 2*PI is negatief, dus voor SIN geldt:



Gaan we meer keren de cirkel rond, dan nemen SIN en COS steeds dezelfde waarden aan als bij de eerste keer:

```
SIN(a+2*PI)=SIN(a)
COS(A+2*PI)=COS(a)
```

De tangens van een hoek wordt gegeven door:

```
TAN(a)=SIN(a)/COS(a)
```

Is de TAN van een hoek bekend, dan kunnen we de hoek afleiden met behulp van ATN (dit staat voor arctan). ATN is de inverse functie van TAN:

```
a=ATN(0.6)
```

geeft de hoek waarvan de TAN gelijk is aan 0.6 en plaatst deze in de variabele a. De hoek wordt gegeven in radialen.
De computer kent geen inverse functie van SIN en COS, maar deze kunt u zelf creëren met behulp van:

```
ARCSIN(a)=ATN(a/SQR(-a*a+1))
ARCCOS(a)=ATN(a/SQR(-a*a+1))+PI/2
```

Een getal dat men vaak in de wiskunde tegenkomt, wordt voorgesteld door de letter e.
Om de eerste 14 cijfers van dit getal te bekijken, moet u intikken:

```
PRINT EXP(1)
```

u krijgt dan:

```
2.7182818284588
```

De functie EXP wordt gegeven als:

```
EXP(x)=e^x
```

dus:

```
EXP(1)=e
```

De inverse van deze functie is LOG. Deze functie geeft u de natuurlijke logaritme van een getal, vaak voorgesteld als LN, die is gerelateerd aan de EXP-functie, zoals hieronder blijkt:

```
EXP(x)=e^x=n
LOG(n)=x
```

U bent wellicht meer gewend aan het gebruik van log's voor het grondtal 10. Om de logaritme voor het grondtal 10 te verkrijgen, moet u de volgende vorm gebruiken:

```
log 10(x)=LOG(x)/LOG(10).
```

Oefeningen

1. Definieer een functie met de naam PI, die de waarde van PI geeft. Gebruik de vorm:

   PI=ATN(1)*4

2. Definieer een functie die graden omzet naar radialen. Gebruik de formule:

   a radialen = (a*PI)/180 graden

   U zult gebruik moeten maken van 3.14... voor PI. Gebruik eventueel de functie PI.

3. Definieer een functie die de ARCCOS geeft van zijn argument. Vergeet echter niet dat geldt: -1 <= COS(a) <= 1, voor iedere hoek van a radialen. Uw functie werkt dus alleen voor waarden tussen -1 en +1.

4. Definieer een functie die de logaritme voor het grondtal 10 geeft van zijn argument.

Hoofdstuk 20

De ASCII-codes

In hoofdstuk 6 heeft u ontdekt dat de letters A-Z en a-z in de computer zijn opgeslagen als achtereenvolgens de nummers 65-90 en 97-122. Deze nummers noemt men ASCII-codes.
ASCII betekent American Standard Code for Information Interchange en wordt gewoonlijk in computers toegepast om tekens voor te stellen.
Als u de ASCII-code van een bepaald teken wilt bepalen, kunt u gebruik maken van de ASC-functie. Bijvoorbeeld:

```
10 A1=ASC("A"):A2=ASC("a")
20 PRINT "De ASCII-code behorend bij A is ";A1
30 PRINT "De ASCII-code behorend bij a is ";A2
```

Het argument van ASC moet een string zijn; indien er meer letters in de string staan, geeft de computer de code van het eerste teken in de string.
De tegengestelde functie van ASC is CHR$. Indien een geschikt numeriek argument wordt meegegeven, dan geeft CHR$ het bijbehorende ASCII-teken terug in een string. Bijvoorbeeld:

```
10 A1$=CHR$(65):A2$=CHR$(97)
20 PRINT "Het teken dat hoort bij ASCII-code 65 is ";A1$
30 PRINT "Het teken dat hoort bij ASCII-code 97 is ";A2$
```

De andere tekens van het toetsenbord hebben ook ASCII-codes. Probeer de ASCII-tekens eens te vinden van '%', '?' en '7'.
In feite bestaan er 256 ASCII-codes, die liggen tussen 0 en 255.
U kunt CHR$ gebruiken om de tekens te vinden die horen bij de codes waarvan de waarde ligt tussen 32 en 255. Draai het volgende programma om deze tekens te vinden:

```
10 REM ASCII-tekens: 32-255
20 CLS
30 FOR I=32 TO 255
40 PRINT CHR$(I)+" ";
50 NEXT I
```

U ziet alle alfanumerieke tekens en codetekens en de meeste grafische

tekens op het scherm verschijnen. Let op dat het laatste teken, dus code 255, de cursor voorstelt; deze verandert als u hem over de tekst heen beweegt!
De codes tussen 0 en 31 zijn speciaal. Zij worden besturingstekens genoemd. Zij stellen geen tekens voor maar bepaalde acties, die de computer dient uit te voeren als ze zijn gebruikt als argument in een PRINT CHR$-opdracht.
U kent reeds het besturingsteken 13, beschreven in hoofdstuk 8. PRINT CHR$ van deze code is equivalent met het indrukken van <RETURN>. Dit is de enige wijze waarop u <RETURN> in een programma kunt gebruiken.
Andere besturingstekens bewegen de cursor, wissen het scherm schoon enz.
Probeer het volgende programma eens:

```
10 REM enkele ASCII-besturingstekens
20 PRINT CHR$(12)+"Besturingsteken met code 12 wist het
scherm schoon."
30 PRINT CHR$(7)+"Besturingsteken met code 7 geeft een
BEEP!"
```

De codes 0-31 bevatten ook de rest van de grafische tekens. Om deze echter te benaderen, moet u eerst een PRINT CHR$(1) doen, en daarna CHR$(A+64), waarin A de bedoelde code is. Het volgende programma drukt de eerste 32 grafische tekens af:

```
10 REM Codes 0-31: meer grafische tekens
20 FOR A=0 TO 31
30 PRINT CHR$(1)+CHR$(A+64)+" ";
40 NEXT A
```

Het sleutelwoord STRING$ wordt gebruikt om een string te creëren van een bepaalde lengte, opgebouwd uit een bepaald teken. De string kan niet langer worden dan 200 tekens, tenzij u een CLEAR-opdracht gebruikt.
STRING$ moet worden voorzien van twee argumenten, gescheiden door een komma en gevat tussen haakjes.
Het eerste argument moet een getal, een numerieke uitdrukking of een numerieke variabele zijn. Deze bepaalt de lengte van de string.
Het tweede argument mag zowel een getal als een string zijn. Indien een getal wordt gebruikt, moet dit de ASCII-code zijn van het in de string af te drukken teken. Bijvoorbeeld:

```
10 FOR B=1 TO 10
20 A$=STRING$(B,42)
30 PRINT A$
40 NEXT B
```

geeft het volgende patroon:

```
*
**
***
****
*****
******
*******
********
*********
**********
```

Is het tweede argument een string, dan wordt het eerste teken van deze string gebruikt, dit is indien regel 20 wordt vervangen door:

```
20 PRINT STRING$(B,"*")
```

zou het resultaat hetzelfde zijn.

Hoofdstuk 21

De verschillende schermen

Tot nu toe heeft u alleen gebruik gemaakt van de tekstschermen, SCREEN 0 en 1. Er bestaan echter nog twee schermen, SCREEN 2 en 3. Deze gaan we gebruiken in de volgende hoofdstukken. Allereerst een samenvatting van enkele eigenschappen van deze schermen.

De tekstschermen

U kunt de breedte van deze schermen veranderen met behulp van het sleutelwoord WIDTH (zie hoofdstuk 9):

SCHERM	Standaard-breedte	Maximum-breedte	Teken-grootte
SCREEN0	37	40	6x8
SCREEN1	29	32	8x8

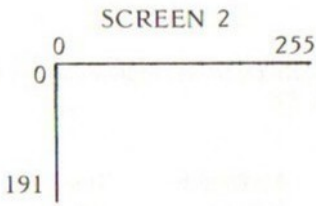
Beide schermen kunnen 24 regels bevatten. Ook al passen er meer tekens over SCREEN 0 dan over SCREEN 1, de hoeveelheid gereserveerde ruimte voor ieder teken is kleiner. Slechts 6x8 punten of pixels zoals ze in het jargon worden genoemd, zijn nodig om één teken te vormen in SCREEN 0, terwijl daarentegen SCREEN 1 8x8 pixels gebruikt. Dit betekent dat de alfanumerieke tekens in SCREEN 0 meer gecomprimeerd verschijnen dan in SCREEN 1, omdat de laatste twee pixelkolommen in de 8x8 matrix die gebruikt wordt bij SCREEN 1, blanco worden gelaten en het zijn juist deze twee kolommen die ontbreken bij de tekenposities van SCREEN 0. Het comprimeren van tekens blijkt duidelijker, wanneer we grafische tekens vergelijken.
Na het inschakelen van de computer is één van de genoemde tekstschermen het standaard-scherm.
De MSX is in staat om 16 verschillende kleuren (zie hoofdstuk 22 voor details) te genereren. Bij SCREEN 1 kunt u de voorgrond (dit is de tekst), de achtergrond en de randen (BORDER) van één van deze 16 kleuren voorzien. U kunt niet tegelijkertijd meerdere voorgrondkleuren gebruiken. De standaardkleuren zijn:

VOORGROND	WIT
ACHTERGROND	DONKERBLAUW
RAND	CYAANBLAUW

Bij SCREEN 0 kunt u alleen de voorgrond- en achtergrondkleur bepalen, waarbij de standaardkleuren gelijk aan de bovengenoemde zijn.
Als u op een willekeurige plaats op het scherm de voorgrondkleur verandert, verandert de kleur van alle tekst op het scherm.

Het HI-RES grafische scherm

SCREEN 2 definieert het HI-RES grafische scherm. HI-RES staat voor HIGH-RESOLUTION en betekent hoog oplossend vermogen.
Dit scherm is 256 pixels breed en 192 pixels lang. Elke pixel kan worden geadresseerd in dit scherm. De pixel linksboven heeft de coördinatenpositie (0,0), en de pixel rechtsonder (255,191):



U kunt de breedte van dit scherm niet veranderen. Gebruikt u een WIDTH-opdracht bij SCREEN 2, dan onthoudt de computer dit; keert u dan terug naar een tekst-SCREEN dan zet de computer de breedte op de tijdens SCREEN 2 ingegeven waarde.
U zou kunnen hebben ingetikt:

```
SCREEN 2
```

terwijl u in de commando-mode was; het zal u dan opvallen dat er niets verandert. Dit komt, omdat u het grafische scherm niet kunt benaderen vanuit de commando-mode. U moet een SCREEN-opdracht meegeven in een programma om het grafische scherm te bekijken, bijvoorbeeld:

```
10 SCREEN 2
20 GOTO 20
```

Regel 20 bevriest het grafische scherm; dit is noodzakelijk, omdat de computer na het uitvoeren van het programma direct naar het standaard-tekstscherm terugkeert. De computer keert ook terug naar het tekstscherm, na een INPUT-opdracht of een foutboodschap.
Om dit programma te beëindigen en terug te keren naar het tekstscherm, moet

u <CNTL> <STOP> indrukken.
De pixels in dit scherm staan samen gegroepeerd in rijen van 8 pixels lang. De eerste pixelrij bevindt zich tussen de coördinaten (0,0) en (8,0).
Een 8x1 pixelrij:

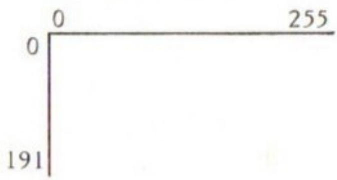


Iedere rij van 8 pixels kan slechts twee kleuren bevatten, een voorgrondkleur en een achtergrondkleur. Aaneengesloten rijen mogen twee verschillende kleuren bevatten. Het kleur-oplossend vermogen van dit scherm is daarom ook 32x192.
In hoofdstuk 14 vindt u hoe u tekens op dit scherm kunt afdrukken. U kunt hiervoor niet direct van de PRINT-opdracht gebruik maken.

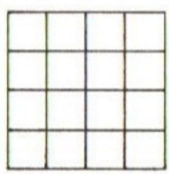
Het meerkleurenscherm

SCREEN 3 staat voor het meerkleurenscherm, en kan worden gebruikt voor grafische weergave met een laag oplossend vermogen (LOW-RES).
Evenals bij SCREEN 2 is dit scherm 256 pixels breed en 192 pixels lang, maar niet iedere pixel kan worden geadresseerd:

SCREEN 3



Het kleinste gebied dat u kunt adresseren op dit scherm, is een blokje van 4x4 pixels. U kunt nog steeds naar (0,0) verwijzen, maar u zult zien dat dan een blok wordt gebruikt dat ook de pixels (1,0), (2,0), (3,0) tot (3,3) beslaat. Verwijzen naar (3,3) zou dus hetzelfde effect hebben.
Een 4x4 pixelblok:



4 x 4 pixelblok

Ieder 4x4 blok kan slechts één kleur bevatten, namelijk een voorgrondkleur. Daarom is het kleuroplossend vermogen van dit scherm 64x48.
Evenals bij scherm 2, kunt u het scherm niet benaderen vanuit de commando-mode; u zult het scherm weer moeten bevriezen met behulp van een oneindige lus. Een INPUT-opdracht of foutboodschap brengt u terug naar het standaard-tekstscherm.

Hoofdstuk 22

Kleur

Na het inschakelen van de computer wordt de achtergrond donkerblauw en de rand er omheen cyaanblauw. De voorgrond of tekst is wit.
U kunt enkele of alle kleuren wijzigen met behulp van het commando COLOR. (Let op de Amerikaanse spelling van dit woord!)
Na ontvangst van het COLOR-commando verwacht de computer drie getallen, gescheiden door komma's. Het eerste getal bepaalt de voorgrondkleur, het tweede de achtergrondkleur en het laatste de kleur van de rand. Er is keuze uit 16 kleuren:

0	transparant
1	blauw
2	middelgroen
3	lichtgroen
4	donkerblauw
5	lichtblauw
6	donkerrood
7	cyaanblauw
8	middelrood
9	lichtrood/roze
10	donkergeel
11	lichtgeel
12	donkergroen
13	roodpaars
14	grijs
15	wit

De standaardkleuren zijn:

```
COLOR 15,4,7
```

De functietoets F6, is voorgeprogrammeerd om COLOR 15,4,7 met een <RETURN> op het scherm af te drukken. Wanneer u per ongeluk de voorgrond dezelfde kleur geeft als de achtergrond, dan kunt u deze toets gebruiken!
Functietoets F1 zet alleen COLOR op het scherm. Het is niet noodzakelijk dat u alle kleurgetallen voor de achtergrond, voorgrond en rand aan COLOR meegeeft. Geef alleen die kleurcode mee die u wilt wijzigen.

Bijvoorbeeld:

```
COLOR 1
```

Dit maakt de voorgrond zwart en laat de achtergrond en rand zoals ze waren.

```
COLOR ,15,15
```

De achtergrond en rand worden veranderd in wit en de voorgrond blijft wit. Let er op dat, ook al is de voorgrondkleur niet aangegeven, u toch een komma dient te specificeren, omdat dit de computer laat weten dat het volgende getal de achtergrond aangeeft.

```
COLOR ,,10
```

Dit commando maakt de rand van het scherm donkergeel. U kunt dit alleen bij de schermen SCREEN 1, 2 of 3 doen, omdat SCREEN 0 geen rand heeft.
Als u de achtergrondkleur wijzigt en u bevindt zich in één van de grafische schermen, dan ziet u het effect hiervan pas na het uitvoeren van CLS. Dit wist het scherm schoon en voorziet de achtergrond van de nieuwe kleur.
Het volgende programma toont alle kleurencombinaties van de rand en de achtergrondkleur die beschikbaar zijn op de MSX. Tik het volgende in met SCREEN 1:

```
10 REM KLEUREN
20 FOR BDR=0 TO 15
30 COLOR,,BDR
40 FOR ACHTER=0 TO 15
50 COLOR,ACHTER
60 FOR VERTRAAG=1 TO 200:NEXT VERTRAAG
70 NEXT ACHTER
80 NEXT BDR
90 COLOR 15,4,7
```

De VERTRAAG-lus in regel 60 biedt de gelegenheid om de verschillende kleuren rustig te observeren; zonder deze regel zouden de kleuren te snel voorbij flitsen, zodat u er niets van waarneemt.
Let er op dat u numerieke variabelen en numerieke uitdrukkingen kunt gebruiken in de COLOR-opdracht. Numerieke uitdrukkingen echter doen niets in COLOR. Als u een reëel getal gebruikt in een COLOR-opdracht, wordt de waarde afgekapt tot de dichtstbijgelegen integere waarde. Het getal moet liggen tussen 0 en 15.
Probeer het programma te draaien in elk van de overige schermen. U moet dan wel een CLS-opdracht toevoegen ten behoeve van de schermen SCREEN 2 en 3. Realiseert u zich wat COLOR 0 doet? Het maakt de kleur van de achtergrond of voorgrond gelijk aan die van de rand. Indien deze wordt gebruikt om de randkleur te bepalen dan wordt de rand zwart.

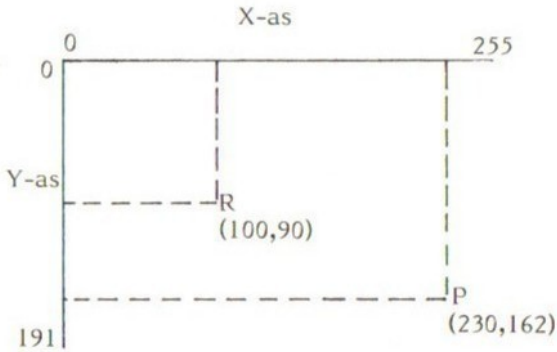
Oefening

1. Programmeer één van de functietoetsen om een andere voorgrond-, achtergrond- en randkleur te verkrijgen.

Hoofdstuk 23

Punten uitzetten

De eerste twee commando's die in dit hoofdstuk worden behandeld, PSET en PRESET, lijken veel op elkaar. Omdat dit grafische commando's zijn, kunnen ze alleen in SCREEN 2 en 3 worden gebruikt.
PRESET wordt gebruikt om een punt in de achtergrondkleur uit te zetten. De coördinaten van deze punt kunnen absoluut of relatief zijn.
Na het gebruik van PRESET zult u zien dat de huidige voorgrondkleur gelijk is aan de achtergrondkleur.
In beide schermen geldt dat de x-coördinaat mag liggen tussen 0 en 255 en de y-coördinaat tussen 0 en 191. Absolute coördinaten worden direct van de x-as en y-as gelezen:



Het punt R bevindt zich op de absolute x-coördinaat 100 en de y-coördinaat 90. P bevindt zich op de coördinaten (230,162). Een relatief coördinatenpaar geeft de positie van een punt, gemeten ten opzichte van een bepaald referentiepunt. Bijvoorbeeld, P bevindt zich op de relatieve coördinaten (130,72), gemeten ten opzichte van R. Dit betekent dat P zich 130 punten verder bevindt dan R langs de x-as en 72 punten lager langs de y-as.
In alle grafische opdrachten wordt de plaats waar de grafische cursor zich het laatst bevindt als referentiepunt genomen.
De positie van de grafische cursor kan worden bepaald met behulp van PRESET:

```
10 SCREEN 2
20 PRESET (100,100)
30 GOTO 30
```

Wanneer u dit programma draait, blijkt er niets te gebeuren! Het is echter zo, dat de grafische cursor zich nu op positie (100,100) bevindt. Ieder grafisch commando dat nu volgt en gebruik maakt van relatieve coördinaten, gebruikt (100,100) als referentiepunt.
Als de volgende opdracht bijvoorbeeld een lijn tekent, dan dient u op te geven welke kleur deze lijn heeft of u moet de kleur van de voorgrond van tevoren aanpassen met de COLOR-opdracht, anders wordt uw lijn onzichtbaar! Indien u een kleur meegeeft in een PRESET-opdracht, wordt een punt uitgezet met deze kleur en wordt diezelfde kleur de nieuwe voorgrondkleur.
Vervang regel 20 door:

```
20 PRESET (100,100),1
```

Het nummer dat volgt op de PRESET moet er een zijn uit de reeks van COLOR-codes, zoals beschreven in hoofdstuk 22, dit is een getal tussen 0 en 15. Wordt een reëel getal gebruikt, dan wordt het decimale gedeelte genegeerd.
Bevinden uw coördinaten zich buiten het scherm, dan is het duidelijk dat u niets ziet! Bevinden de coördinaten zich buiten de toegestane integere waarden -32768 en 32767, dan volgt een foutboodschap.
Het commando PSET zet een punt uit in de voorgrondkleur, op de aangegeven coördinaten. Deze coördinaten mogen absoluut en relatief zijn.
Geeft u een kleurcode mee, dan wordt het punt in deze kleur afgedrukt, zodat PSET dus gelijk is aan PRESET met een kleurcode. Evenals bij PRESET wijzigt dit commando de voorgrondkleur in de aangegeven kleur.
Het volgende programma zet willekeurig gekleurde blokken uit, op scherm SCREEN 3, met behulp van PSET. Deze blokken worden vervolgens uitgewist door het terugzetten van dezelfde willekeurige punten in de achtergrondkleur, met behulp van PRESET:

```
10 REM STIPPEN
20 R=TIME
30 RDRST=RND(-R)
40 SCREEN 3
50 REM DRUK WILLEKEURIG GEKLEURDE STIPPEN AF
60 FOR I%=1 TO 300
70 GOSUB 1000
80 PSET(X%,Y%),Z%
90 NEXT I%
100 REM ZET STIPPEN TERUG NAAR ACHTERGRONDKLEUR
110 RDRST=RND(-R)
120 FOR J%=1 TO 300
130 GOSUB 1000
140 PRESET (X%,Y%)
150 NEXT J%
990 REM SUBROUTINE
```

```
1000 X%=RND(1)*256
1010 Y%=RND(1)*192
1020 Z%=RND(1)*16
1030 RETURN
```

Dit programma gebruikt integere variabelen om de snelheid op te voeren, dit is X% en deze wordt gebruikt in plaats van X.
Iedere keer als het programma wordt gestart, worden de punten op andere willekeurige plaatsen uitgezet. Dit wordt bereikt met behulp van de variabele TIME, die de reeks van punten bepaalt. TIME heeft iedere keer als u het programma draait een andere waarde. Dezelfde getallen worden in het tweede gedeelte van het programma gebruikt, zodat kan worden gewist. Om dezelfde willekeurige getallen te verkrijgen, zet u de generator van willekeurige getallen terug, door eenzelfde negatief argument mee te geven aan RND in regel 110, evenals in regel 30.
Ten slotte beschrijven we nog het **POINT**-commando. Dit is een ander grafisch commando. Het geeft u de kleur van een punt terug, in één van de grafische schermen. Het teruggegeven getal ligt daarom tussen 0 en 15, en geeft daarmee een kleurcode aan van transparant tot wit. -1 wordt teruggegeven als de meegegeven coördinaten zich buiten het scherm bevinden. Tik in:

```
10 COLOR ,2
20 SCREEN 2
30 P=POINT(100,100)
```

Eenmaal terug in commando-mode, tik in:

```
PRINT P
```

Het getal 2 wordt getoond, omdat dit de kleurcode is van middelgroen, wat de nieuwe achtergrondkleur is.

Oefening

1. Probeer een SINUS-curve uit te zetten, met behulp van:

   ```
   Y=A*SIN(X)+A
   ```

   A bepaalt de amplitude van de curve. Laat X variëren tussen 0 en 2*PI.

Hoofdstuk 24

Het tekenen van lijnen en vierkanten

Dit hoofdstuk behandelt het LINE-commando. Dit is weer een grafisch commando en kan dus alleen worden gebruikt in de schermen SCREEN 2 en 3.

Het trekken van lijnen

Bij het trekken van een lijn kunt u het start- en eindpunt aangeven, met behulp van absolute of relatieve waarden of zelfs beide tegelijk. Tik het volgende programma in:

```
10 SCREEN 2
20 LINE (0,0)-(255,191)
30 GOTO 30
```

Dit programma trekt een lijn als diagonaal over het scherm. Let op het teken '-' tussen de coördinatenparen.
Er is gebruik gemaakt van absolute coördinaten. Het eerste coördinatenpaar (0,0) geeft de start aan van de lijn; (255,191) het eindpunt.
Wijzig regel 10 van dit programma, opdat de lijn in SCREEN 3 wordt getrokken, en vergelijk het oplossend vermogen van de twee grafische schermen.
Probeer ook verschillende coördinatenparen in het LINE-commando.
Is de door u opgegeven waarde van de x- of y-coördinaat groter dan respectievelijk 255 en 191, dan trekt de computer een lijn tot aan de rand van het scherm, stelt x op 255 en y op 191. Gebruikt u waarden buiten de toegestane reeks, dus buiten -32768 en 32767, dan volgt een foutboodschap.
U hoeft niet altijd het startpunt mee te geven aan LINE. Specificeert u geen startpunt, dan start de computer vanaf de huidige cursorpositie. Had u dus als laatste een LINE-commando opgegeven, dan bevindt de cursor zich dus aan het eind van de getrokken lijn.
U zou ook relatieve coördinaten mee hebben kunnen geven, om dezelfde lijn te trekken. Vervang regel 20 van het programma door:

```
20 LINE (0,0)-STEP (255,191)
```

Het STEP-commando geeft aan dat de navolgende coördinaten als relatief

dienen te worden beschouwd door de computer. De uiteindelijke positie van x zal dus 255 posities verder liggen dan het beginpunt, de uiteindelijke positie van y 191 posities verder; de cursor is nu op de absolute coördinaten (255,191) gezet (dit is het laatste punt van LINE).

Gekleurde lijnen

Tot nu toe heeft u alleen lijnen getrokken in de voorgrondkleur. Het is mogelijk om de kleur aan te geven van de lijn, door het LINE-commando te voorzien van een komma en een geldige kleurcode, tussen 0 en 15. Zie hoofdstuk 22 voor de COLOR-codes. Vervang nu regel 20 van het programma door:

```
10 LINE -(255,191),10
```

Dit tekent een gele lijn diagonaal over het scherm.
Het volgende programma illustreert de kleuroplossing in SCREEN 2.

```
10 SCREEN 2
20 FOR A= 0 TO 15
30 LINE (0+A,0)-(200+A,191),A
40 NEXT A
50 GOTO 50
```

U krijgt 16 diagonale lijnen te zien over het scherm, maar de kleuren zijn vlekkerig. Als u regel 30 vervangt door:

```
30 LINE (0,0+A)-(255,0+A),A
```

bent u in staat om iedere kleur van de lijnen te zien. U krijgt deze effecten door de wijze waarop het scherm is opgeslagen in het geheugen van de computer. Iedere horizontale regel is opgedeeld in groepen van elk acht pixels lang. Iedere groep mag hooguit twee kleuren per keer bevatten, namelijk een voorgrond- en een achtergrondkleur. Na het plaatsen van de eerste diagonale lijn wordt de voorgrondkleur op 0 gezet en wordt de lijn dus in deze kleur getrokken. De volgende lijn zet de voorgrondkleur op 1, maar dit verandert dus ook de kleur van de groep van acht pixels, waardoor de eerste pixel van de eerste lijn zwart wordt. De achtste lijn verandert de voorgrondkleur van de eerste groep van acht pixels voor de laatste keer; de gehele groep is nu cyaanblauw.
De verticale kleuroplossing is iets beter dan de horizontale. De kleur van iedere verticale pixel is onafhankelijk van de pixels erboven en eronder, vandaar dat de 16 horizontale lijnen elkaar niet beïnvloeden.
Probeer de twee programma's te draaien in SCREEN 3. In dit scherm ziet u steeds dezelfde vier lijnen, omdat de horizontale en verticale kleuroplossing dezelfde zijn. U kunt een kleur hebben voor ieder 4x4 pixelblok. De

kleuroplossing is daarom gelijk aan de grafische oplossing. De eerste vier lijnen worden boven op elkaar geplaatst, zodat u alleen de laatste te zien krijgt, die lichtgroen is.

Het tekenen van rechthoeken

Het is mogelijk om een rechthoek te tekenen met behulp van vier achtereenvolgende LINE-opdrachten. Er is echter een eenvoudiger manier om rechthoeken te tekenen, als u de coördinaten weet van de linkerbovenhoek en de rechteronderhoek. U gebruikt wel het LINE-commando, maar nu zijn de eerste coördinaten de linkerbovenhoek en de tweede de rechteronderhoek. Om de computer te laten weten dat u een rechthoek wilt hebben, en niet een lijn, moet u achter de komma met de kleurcode, nog een komma plaatsen met de letter B van BOX. Tik het volgende programma in:

```
10 REM RECHTHOEKEN
20 SCREEN 2
30 FOR A=0 TO 90 STEP 10
40 LINE (5+A,5+A)-(255-A,185-A),A/10,B
50 NEXT A
60 GOTO 60
```

Dit programma tekent tien rechthoeken met verschillende maten en kleuren.
Let er op dat u een uitdrukking kunt gebruiken als kleurcode, maar deze uitdrukking moet als resultaat een waarde voorstellen tussen 0 en 15. Als het resultaat een reëel getal voorstelt, dan wordt het decimale gedeelte genegeerd.
Let ook eens op het effect van dit programma in SCREEN 3, door het wijzigen van regel 20:

```
20 SCREEN 3
```

Als u de lengte weet van de rechthoek, en niet de coördinaten van de hoeken, dan is het eenvoudiger om relatieve coördinaten te gebruiken voor het tweede coördinatenpaar, dit is als u een rechthoek wilt tekenen van 100 pixels lang in de x-richting en 50 pixels in de y-richting, dan kunt u het volgende doen:

```
10 SCREEN 2
20 LINE (10,10)-STEP(100,50),,B
30 GOTO 30
```

Het programma tekent een rechthoek met de juiste dimensies in de huidige voorgrondkleur. De rechterbovenhoek is gesitueerd op de absolute positie (10,10).
Het tweede coördinatenpaar specificeert de lengte van de zijden van de rechthoek, die parallel aan de x- en de y-as zijn.
Dit is de aangewezen methode om rechthoeken te tekenen. Bijvoorbeeld:

```
10 REM RECHTHOEKEN
20 SCREEN 2
30 FOR A=10 TO 90
40 LINE (A,A)-STEP(A,A),,B
50 NEXT A
60 GOTO 60
```

Het laatste programma tekent rechthoeken, met toenemende grootte, diagonaal over het scherm.

Gekleurde rechthoeken

Om een rechthoek te tekenen en te vullen met een opgegeven kleur, dient u de letter 'B' te vervangen door 'BF'. De kleur die ook gespecificeerd staat in de LINE-opdracht wordt nu gebruikt om de rechthoek te kleuren:

```
10 REM zwarte doos
20 SCREEN 2
30 LINE (20,20)-(100,100),1,BF
40 GOTO 40
```

Oefening

1. Schrijf een programma om een doos te tekenen van een willekeurige grootte, ergens willekeurig op het scherm en ingevuld met een willekeurige kleur.

Hoofdstuk 25

Het tekenen van cirkels en ellipsen

Het CIRCLE-commando stelt u in staat een hele, of een deel van een cirkel of ellips te tekenen. U moet aangeven wat de positie van het centrum van de cirkel is, met absolute of relatieve coördinaten, en de lengte van de straal. U heeft ook de mogelijkheid om de cirkel van een kleur te voorzien en om aan te geven welk gedeelte van de cirkel u wilt afdrukken. CIRCLE is weer een grafisch commando en kan daarom alleen bij de grafische schermen worden gebruikt.

Het tekenen van cirkels

Om een cirkel te tekenen, dient u de coördinaten van het middelpunt en de lengte van de straal te specificeren. Bijvoorbeeld:

```
10 REM cirkels tekenen
20 SCREEN 2
30 CIRCLE (128,100),90
40 GOTO 40
```

Dit programma tekent een cirkel met het absolute middelpunt (128,100) en een straal van 90 pixels, in de huidige voorgrondkleur. Het zal duidelijk zijn dat, met een voor het beeldscherm te grote straal, u geen cirkel te zien krijgt, in ieder geval geen volledige.
Om een kleur te specificeren voor de cirkel, laat u de straallengte volgen door een komma en de gewenste kleurcode, dit is een waarde tussen 0 en 15. (Zie hoofdstuk 22 voor de kleurcodes.) Het volgende programma tekent dezelfde cirkel, maar nu in zwart:

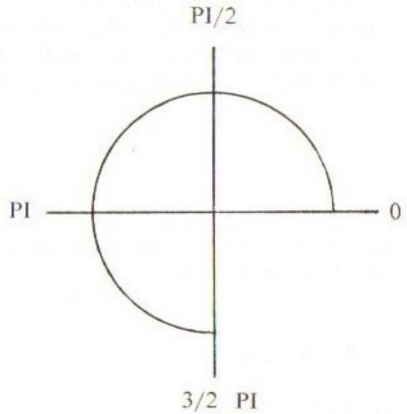
```
10 SCREEN 2
20 CIRCLE (196,100),90,1
30 GOTO 30
```

Het tekenen van een cirkelboog

Om alleen een gedeelte van een cirkel of cirkelboog te tekenen, dient u de begin- en eindhoeken te specificeren. Het volgende programma tekent driekwart van een zwarte cirkel:

```
10 REM tekenen van cirkelbogen
20 SCREEN 2
30 PI=4*ATN(1)
40 CIRCLE (128,100),90,1,0,3*PI/2
50 GOTO 50
```

De laatste twee getallen die we hebben toegevoegd, geven de begin- en eindhoek van de cirkelboog. De hoeken moeten worden gespecificeerd in radialen. (Als u niet meer weet wat een radiaal is, kijk dan in hoofdstuk 19.) Regel 30 berekent PI, zodat de boog wordt getekend tussen de hoeken 0 en 3/2 PI radialen:



Alle hoeken worden gemeten vanaf de X-as, tegen de beweging van de wijzers van de klok in. Omdat er 2*PI radialen in een cirkel zijn, kan de opgegeven waarde alleen tussen 0 en 2*PI liggen.

Het tekenen van een punt uit een cirkel

Om dit te bereiken moet u het teken '-' plaatsen voor de begin- en eindhoeken. Dit heeft tot gevolg dat er lijnen worden getrokken vanuit het centrum van de cirkel naar het begin- en eindpunt van de cirkelboog. Draai het volgende programma:

```
10 REM tekenen van een punt
20 SCREEN 2
30 PI=4*ATN(1)
40 CIRCLE (128,100),90,,-PI/2,-PI
50 GOTO 50
```

U krijgt nu de linker bovensector van de cirkel te zien, in de voorgrondkleur.

Het tekenen van ellipsen

Om een ellips te tekenen, moet u de aspectratio meegeven. De aspectratio wordt gedefinieerd als:

$$\text{ASPECTRATIO} = \frac{\text{verticale straal}}{\text{horizontale straal}}$$

Als de aspectratio groter is dan 1, wordt de ellips verticaal verlengd. De gespecificeerde straal wordt genomen als de verticale straal.
Als de aspectratio kleiner is dan 1, wordt de ellips horizontaal verlengd, en de gespecificeerde straal als de horizontale straal genomen.
Het volgende programma tekent een complete zwarte ellips, omdat geen begin- en eindhoek zijn meegegeven:

```
10 REM tekenen van ellipsen
20 SCREEN 2
30 CIRCLE (128,100),90,1,,,1.5
40 GOTO 40
```

Omdat de aspectratio gelijk is aan 1.5 wordt de ellips verticaal verlengd en is de verticale straal 90 pixels.
Probeer aspectratio's 100 en 0.01. Met deze aspectratio's verschijnen de ellipsen als respectievelijk verticale en horizontale lijnen, met een lengte van 180 pixels.
Het volgende programma tekent verschillende gekleurde ellipsen met willekeurige aspectratio's, allen in het midden van het scherm gecentreerd:

```
10 REM elliptische patronen
20 SCREEN 2
30 FOR B=1 TO 15
40 A=RND(-TIME)*2
50 CIRCLE (128,96),90,B,,,A
60 NEXT B
70 GOTO 70
```

Als u dat wilt, kunt u een gedeelte van de ellips tekenen door het meegeven van de begin- en eindhoek, evenals dit gebeurde bij de eerder behandelde cirkels. Het teken '-' kan ook worden gebruikt:

```
10 SCREEN 3
20 PI=4*ATN(1)
30 FOR B=1 TO 15
```

```
40 A=B/10
50 SA=RND(-TIME)*2*PI
60 EA=RND(-TIME)*2*PI
70 CIRCLE (126,96),90,B,SA,EA,A
80 NEXT B
90 GOTO 90
```

Het laatste programma tekent waaiervormen, gecentreerd op de coördinaten (126,96). De begin- en eindhoeken worden berekend met behulp van de RND-functie. Let er op dat de eindhoek kleiner kan zijn dan de beginhoek. De aspectratio wordt gevarieerd van 0.1 tot 1.5. Voer het programma uit in scherm SCREEN 2, om zodoende het oplossend vermogen te vergelijken.

Oefeningen

1. Schrijf een programma om tien kleine cirkels of ellipsen willekeurig op het scherm te tekenen.

2. Verander het programma zo, dat ze allen verschillende stralen hebben.

3. Teken een cirkel die is opgebouwd uit vier sectoren van verschillende kleuren.

Hoofdstuk 26

De grafische macrotaal

De grafische macrotaal simuleert de bewegingen van een pen op papier en stelt u in staat op eenvoudige wijze gecompliceerde beelden te tekenen. Met behulp van het DRAW-commando kunt u de cursor naar een willekeurige plaats op het beeldscherm verplaatsen, met de pen omhoog of omlaag.

Het DRAW-commando dient te worden gevolgd door een string. Deze string bevat de grafische macrocommando's die worden voorgesteld door enkelvoudige tekens.

Om een lijn te trekken in één van de vier richtingen omhoog, omlaag, rechts of links, worden de volgende grafische commando's gebruikt:

U	tekent omhoog (UP)
D	tekent omlaag (DOWN)
L	tekent links (LEFT)
R	tekent rechts (RIGHT)

U moet de lengte van de lijn meegeven in het commando. De lengte wordt gemeten in pixels.

Het volgende programma demonstreert het gebruik van de macrotaal:

```
10 REM vierkant
20 SCREEN 2
30 PSET (123,91)
40 DRAW "R10D10L10U10"
50 GOTO 50
```

Na het zien van regel 40 voert de computer de macrocommando's achtereenvolgens uit, dus trekt een lijn naar rechts, met lengte 10, daarna 10 pixels omlaag, 10 naar links en ten slotte tien omhoog, zodat het vierkant klaar is.

Met de volgende vier commando's kunnen diagonale lijnen worden getrokken:

E	tekent diagonaal rechts omhoog
F	tekent diagonaal rechts omlaag
G	tekent diagonaal links omlaag
H	tekent diagonaal links omhoog

Opnieuw dient ieder commando te worden gevolgd door een getal. In dit geval geeft het getal echter aan, hoeveel pixels in de x- en y-richting moet worden geschoven.
Bijvoorbeeld, zullen de relatieve coördinaten van het laatste punt van de lijn getekend door E10, gelijk aan (10,10) zijn, met als referentiepunt het begin van de lijn.
Het volgende programma tekent een zeshoek:

```
10 REM zeshoek
20 SCREEN 2
30 PSET (128,96)
40 DRAW "E20F20D30G20H20U30
50 GOTO 50
```

Om een lijn te trekken naar een specifiek punt op het scherm, wordt het M-commando gebruikt. Dit commando moet worden gevolgd door respectievelijk de x- en y-coördinaat van het laatste punt van de lijn. De x- en y-waarden mogen zowel relatief als absoluut zijn. Bijvoorbeeld:

```
M40,50
```

verplaatst de cursor naar de absolute positie (40,50), terwijl de lijn wordt getekend.
Om relatieve coördinaten mee te geven, wordt gebruik gemaakt van de prefix + of -:

`M+40,50`	Tekent naar een positie, +40 pixels langs de x-as en +50 langs de y-as.
`M+40,-50`	Tekent naar een positie, +40 pixels langs de x-as, maar -50 pixels langs de y-as.

Let er op dat de volgende commando's gelijk zijn aan elkaar:

```
M+0,-10=U10          M+10,-10=E10
M+0,10=D10           M+10,10=F10
M+10,0=R10           M-10,10=G10
M-10,0=L10           M-10,-10=H10
```

In het laatste programma werd PSET gebruikt om de plaats van de grafische cursor te bepalen, nog voordat werd getekend. Dit kan in feite ook worden gedaan met behulp van de prefix B voor het M-commando. B staat voor Blanco. Deze prefix zorgt er voor dat de cursor wordt verplaatst zonder daarbij te tekenen.

```
30 DRAW "BM128,96"
```

zou kunnen worden gebruikt om de cursor in het centrum van het scherm te plaatsen.

De B-prefix mag ook bij de eerder genoemde macrocommando's worden gebruikt, hetgeen waarschijnlijk is omdat dit allemaal speciale uitvoeringen zijn van het M-commando.
Indien u wenst terug te keren naar de startpositie, na het tekenen van een lijn, gebruikt u de prefix N.
Het volgende programma tekent een ster op uw scherm:

```
10 REM ster
20 SCREEN 2
30 REM verplaats cursor
40 DRAW "BM128,96"
50 REM teken armen
60 DRAW "NU50ND50NR50NL50"
70 DRAW "NE30NF30NG30NH30"
100 GOTO 100
```

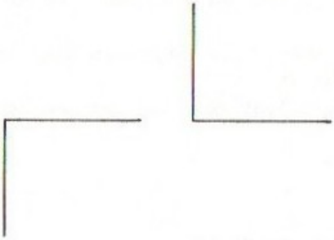
Voor iedere te tekenen arm keert de cursor terug naar het centrum van de ster.
U zou nu in staat moeten zijn om redelijk gecompliceerde vormen te tekenen. Zodra u uw figuur heeft getekend, is het mogelijk om de oriëntatie ervan op het scherm te wijzigen met behulp van het hoekcommando A.
Dit commando laat de assen van het scherm roteren tegen de richting in van de beweging van de wijzers van de klok, dus tot 0, 90, 180 of 270 graden, vanaf de normale standaard-oriëntatie. Dit op zijn beurt beïnvloedt de oriëntatie van de commando's U, D, L, R, F, E, G en H.
Het A-commando moet worden gevolgd door een integer getal tussen 0 en 3, waarin:

An	ROTATIE
A0	zet de assen terug in de normale situatie.
A1	roteert de assen 90 graden tegen de klok in.
A2	roteert de assen 180 graden tegen de klok in.
A3	roteert de assen 270 graden tegen de klok in.

Zodra A is uitgevoerd, zal de oriëntatie van alle volgende commando's worden geroteerd met de opgegeven hoek. Om terug te keren naar de normale situatie, moet u A0 gebruiken binnenin een DRAW-opdracht.
Voer het volgende programma tweemaal uit:

```
10 REM hoek
20 SCREEN 2
30 DRAW "BM50,100NR50ND50"
40 DRAW "A1BM150,100NR50ND50"
50 GOTO 50
```

De eerste keer ziet u:



De tweede keer:



Dit komt omdat het A-commando nog steeds op A1 staat. Als u het eerste resultaat wenst te herhalen, dient u de rotatiehoek terug te zetten op A0 aan het begin van de string in regel 30.
Tot nu toe zijn alle tekeningen in de huidige voorgrondkleur getekend. Om in een andere kleur te tekenen, dus om de voorgrondkleur te veranderen, kunt u het C-commando gebruiken. Dit commando moet worden gevolgd door een geldige kleurcode. De kleurcodes zijn gelijk aan die van het COLOR-commando, zoals behandeld in hoofdstuk 22. Dus:

C0	transparant	C8	middelrood
C1	zwart	C9	lichtrood
C2	middelgroen	C10	donkergeel
C3	lichtgroen	C11	lichtgeel
C4	donkerblauw	C12	donkergroen
C5	lichtblauw	C13	roodpaars
C6	donkerrood	C14	grijs
C7	cyaanblauw	C15	wit

Evenals bij het hoekcommando geldt dat, zodra C is gezet, de voorgrondkleur zo blijft, totdat een nieuw C-commando of COLOR-commando volgt.
Het S-commando verandert de schaal of tekengrootte. Het getal n, dat volgt op S, geeft de nieuwe schaal aan. Dit getal mag iedere waarde aannemen tussen 0 en 255.
De schaalfactor (SF) wordt gedefinieerd als:

SF = n/4

Daarna resulteert S1 in een SF van 1/4. Alle lengtes die zijn aangegeven bij de volgende U-, D-, L- en R-commando's enz. zijn op basis van een

schaal van 1/4. Voer het volgende programma uit:

```
10 REM schaal
20 SCREEN
30 DRAW "BM50,50"
40 DRAW "R150D100L150U100"
50 DRAW "BM50,50"
60 DRAW "S1R150D100L150U100"
70 GOTO 70
```

U ziet twee rechthoeken, de grootte van de ene is éénvierde van de grootte van de andere. Als u het programma opnieuw uitvoert, worden beide rechthoeken op elkaar getekend, met de verkleinde schaal.
Om de SF terug te zetten op 1, mag u zowel S4 als S0 gebruiken. Voeg één van deze twee instructies toe aan het begin van de string in regel 40 en voer het programma nogmaals uit.
Als u een vorm meer dan één keer wenst te tekenen in een programma, dan is het aan te raden om de stringgegevens in een stringvariabele op te slaan. Het volgende programma tekent vier diamanten op het scherm:

```
10 REM DIAMANTEN
20 SCREEN 2
30 A$="F40G40H40E40"
40 DRAW"BM50,8"
50 DRAW A$
60 DRAW"BM200,8"
70 DRAW A$
80 DRAW"BM50,104"
90 DRAW A$
100 DRAW"BM200,104"
110 DRAW A$
120 GOTO 120
```

Het programma kan aanzienlijk worden ingekort, door meerdere commando's op te nemen in het DRAW-commando. Om een stringvariabele mee te geven aan een DRAW-opdracht, dient u het X-commando te gebruiken. Het volgende programma tekent ook vier diamanten, maar dit keer van verschillende grootte. De stringvariabele A$ wordt gebruikt als sub-string:

```
10 REM DIAMANTEN OP SCHAAL
20 SCREEN 2
30 A$="F40G40H40E40"
40 DRAW"S4BM50,8XA$;"
50 DRAW"S3BM200,8XA$;"
60 DRAW"S2BM50,104XA$;"
70 DRAW"BM200,104XA$;"
80 GOTO 80
```

Let er op dat de stringvariabele wordt gevolgd door een puntkomma, dit mag men niet vergeten.
Als u een numerieke variabele wilt gebruiken in een DRAW-string, bijvoorbeeld om de lengte van een lijn te bepalen, dan moet u de naam van de variabele vooraf laten gaan door '='.
Opnieuw moet de naam worden gevolgd door een puntkomma. Het volgende programma tekent 15 dozen en gebruikt daarvoor de variabele I, om de schaal en de kleur van iedere doos te bepalen:

```
10 REM groeiende dozen
20 SCREEN 2
30 DRAW "BM31,186"
40 FOR X=1 TO 15
50 DRAW "S=X;C=X;U48R48D48L48"
60 NEXT X
70 GOTO 70
```

De verticale linkerzijden van de dozen zijn zorgvuldig gepositioneerd, om precies op de laatste van een horizontale groep van 8 pixels terecht te komen. Dit is gedaan om te voorkomen dat de laatste verticale witte lijn en de 14 andere horizontale lijnen die de bovenkanten van de andere dozen bepalen, gaan vlekken. Het vlekeffect ontstaat als u twee kleuren probeert te gebruiken in een horizontale groep van 8 pixels. (Zie ook hoofdstuk 21 voor verdere gegevens.) Om het vlekeffect te zien, kunt u regel 30 vervangen door:

```
30 DRAW "BM25,186"
```

Oefeningen

1. Gebruik de S- en C-commando's om sterren willekeurig op het scherm te tekenen, ieder met een verschillende kleur en grootte.

2. Teken een huis.

Hoofdstuk 27

Inkleuren (PAINT)

Het commando PAINT kan worden gebruikt om iedere figuur die is omgeven door een grenslijn te vullen met een bepaalde kleur. De coördinaten in het PAINT-commando moeten een plaats aangeven binnen de te kleuren vorm. Deze kunnen zowel relatief als absoluut zijn.
De voorwaarden van de toegestane kleuren verschillen in de beide grafische modes.

SCREEN 2

Voordat u PAINT kunt gebruiken, moet u eerst de figuur tekenen met een van de grafische commando's, LINE, CIRCLE of DRAW. In dit scherm moet de PAINT-kleur gelijk zijn aan de kleur die stond aangegeven in het grafische commando. Bijvoorbeeld:

```
10 REM GEKLEURD PARALLELOGRAM
20 SCREEN 2
30 PRESET (100,100)
40 LINE -STEP(-50,50),6
50 LINE -STEP(100,0),6
60 LINE -STEP(50,-50),6
70 LINE -STEP(-100,0),6
80 PAINT(110,110),6
90 GOTO 90
```

De coördinaten (110,110) bevinden zich in het parallelogram. De vorm van een rood parallelogram wordt getekend en daarna gevuld met de rode PAINT. Tracht u echter het parallelogram met een andere kleur te vullen, dan loopt het PAINT-commando buiten de figuurgrens en vult het hele scherm.
Als de figuur zou zijn getekend in de huidige voorgrondkleur, dan zou het niet nodig zijn geweest om een kleur aan te geven in het PAINT-commando, omdat wanneer geen kleur wordt meegegeven in deze opdracht, de huidige voorgrondkleur wordt gebruikt. Bijvoorbeeld:

```
10 REM gekleurde cirkel
20 SCREEN 2
```

```
30 COLOR 10
40 CIRCLE (100,100),50
50 PAINT STEP (0,0)
70 GOTO 70
```

Regel 40 zet de voorgrondkleur op geel. De cirkel wordt daarom in geel getekend en daarna gevuld met geel.
Let er op dat de PAINT-coördinaten gelijk zijn aan het middelpunt van de cirkel.
In SCREEN 2 moet men zien te voorkomen dat het PAINT-commando tot vlekkerige resultaten leidt. Het volgende programma tekent twee driehoeken die tegen elkaar aan staan, om zodoende een vierkant te vormen. Eén van de twee driehoeken wordt dan met PAINT geel gemaakt, de ander zwart.

```
10 REM DRIEHOEKEN V1
20 SCREEN 2
30 COLOR 10
40 PSET(48,50)
50 DRAW"D96R96H96"
60 PAINT (60,70)
70 COLOR 1
80 PSET (70,60)
90 DRAW"R96D96H96"
100 PAINT(70,60)
110 GOTO 110
```

Als u dit programma start, krijgt u een zig-zag-effect. Dit komt omdat in SCREEN 2 iedere horizontale groep van 8 pixels niet meer dan twee kleuren kan bevatten. Er is geen probleem als de gele driehoek wordt gekleurd, want de voorgrond is geel en de achtergrond is blauw (de normale kleur). Als de figuur van de zwarte driehoek wordt getekend, is echter de achtergrondkleur nog steeds blauw, maar de nieuwe voorgrondkleur zwart. Het meeste van de gele driehoek wordt niet beïnvloed, maar daar waar een groep van 8 pixels in beide driehoeken ligt, wordt de voorgrondkleur van de complexe groep in zwart veranderd. Dit heeft tot gevolg dat kleine stukjes van de gele driehoek opgaan in de zwarte driehoek, vandaar het zig-zag-effect.
Er is een eenvoudige oplossing voor dit probleem. Voer het volgende programma uit:

```
10 REM DRIEHOEKEN V2
20 SCREEN 2
30 COLOR 10
40 PSET(48,50)
50 DRAW"D96R96U96L96"
60 PAINT (60,70)
70 COLOR 1
80 PSET (48,50)
90 DRAW"R96D96H96"
100 PAINT(70,60)
110 GOTO 110
```

In dit programma wordt een geel vierkant getekend en geel gekleurd. Daarna wordt een zwarte driehoek getekend en gekleurd, bovenop het vierkant. Omdat het vierkant maar één kleur bevat, is er geen probleem bij het tekenen van de zwarte driehoek.

SCREEN 3

In dit scherm kunt u een vorm kleuren en daarna een rand er omheen zetten:

```
10 REM kleuren met een rand
20 SCREEN 3
30 LINE (20,20)-(100,100),7,B
40 PAINT (50,50),9,7
50 GOTO 50
```

Er worden twee getallen meegegeven aan PAINT. Het eerste getal geeft de kleur aan - dit mag iedere kleur zijn. Het tweede getal geeft de kleur van de rand aan. De kleur van de rand moet gelijk zijn aan de kleur van de grenslijn van de figuur. In bovenstaand voorbeeld was de doos oorspronkelijk in cyaanblauw getekend, daarom moet de randkleur ook cyaan worden. Probeert u een van de andere kleuren voor de rand, dan loopt PAINT buiten de grenzen van de figuur en vult het gehele scherm.
Was de figuur in de kleur van de huidige voorgrond getekend, dan dient u de randkleur dezelfde te laten zijn als u een andere kleur specificeert voor de rest van de figuur. Voer het volgende programma uit:

```
10 REM DRIEHOEK
20 SCREEN 3
30 COLOR 15
40 LINE (20,20)-STEP(200,100)
50 LINE -STEP(-200,0)
60 LINE -(20,20)
70 PAINT (50,80),10,15
80 GOTO 80
```

U ziet een driehoek in wit getekend worden, die daarna geel wordt gekleurd en voorzien van een witte rand. Had u geen witte kleur voor de rand opgegeven in het bovenstaande programma, dan had PAINT het gehele scherm gekleurd.
Wilt u echter een compleet witte driehoek, vervang dan regel 70 door:

```
70 PAINT (50,80)
```

Oefeningen

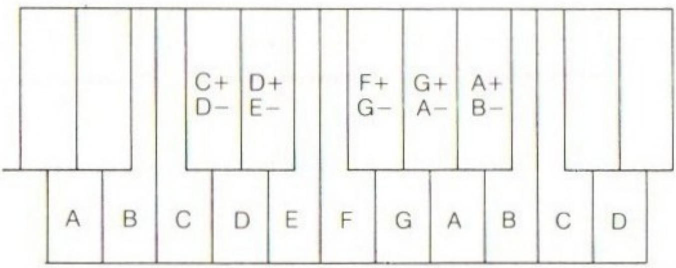
1. Schrijf een programma dat een zwarte doos op het scherm tekent, met behulp van de 'BF'-mogelijkheid in de LINE-opdracht. Teken een andere zwarte doos ernaast, met behulp van de 'B'-mogelijkheid in de LINE-opdracht en PAINT. Draai dit programma in SCREEN 2 en 3.

2. Schrijf een programma in SCREEN 2 dat meerkleuren concentrische cirkels tekent, waarbij iedere cirkel met een andere kleur wordt gevuld. Vergeet niet het vlekkerige effect te voorkomen en begin met de grootste cirkel.

Hoofdstuk 28

De muziek-macrotaal

Wanneer u bekend bent met de standaard-muzieknotatie, dan kunt u met behulp van het PLAY-commando zeer eenvoudig muziek schrijven of overnemen uit een manuscript.
De muziek-macrotaal lijkt veel op de grafische macrotaal, die u bent tegengekomen in hoofdstuk 26. Het sleutelwoord PLAY wordt gevolgd door een string die de verschillende commando's bevat voor de muziek-macrotaal.
De commando's A, B, C, D, E, F en G spelen de bijbehorende noten. Ieder van deze commando's kan worden gevolgd door een '+'- of '-'-teken om aan te geven of de bijbehorende toon dur of mol is. De dur- of mol-noot moet echter altijd corresponderen met een zwarte toets op de piano; B+ is daarom fout.



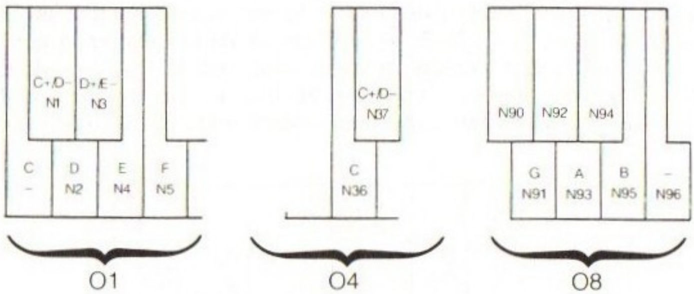
We hebben nu de mogelijkheid om noten te maken in een octaaf. Om de gewenste octaaf aan te geven, gebruiken we het O-commando. De eerste noot in een octaaf is C, de laatste B. Het octaaf-commando moet worden gevolgd door een getal dat ligt tussen 1 en 8. De beginwaarde voor O is 1, waarin de begintoon, de C, correspondeert met de middelste C-toets op de piano. Met behulp van O1 tot O8 heeft u toegang tot iedere noot die u kunt spelen op een piano.
Zodra de octaafwaarde is gezet, wordt voor alle noten de octaafwaarde aangepast die in het programma staat. Het volgende programma speelt E-majeur (E-dur) over twee octaven:

```
10 REM E-majeur: A-G notatie
20 PLAY "EF+G+ABO5C+D+EF+G+ABO6C+D+E"
```

Indien u dit programma opnieuw uitvoert, wordt begonnen vanaf het zesde octaaf. Om dit te voorkomen, moet u een O4-commando toevoegen aan het begin van de string. Zodoende wordt regel 20:

```
20 PLAY "O4EF+G+ABO5C+D+DE+G+ABO6C+D+E"
```

Iedere toon binnen de 8 octaven kan ook worden gespeeld met behulp van het N-commando. Het commando moet worden gevolgd door een getal tussen 0 en 96. Omdat er 96 noten zijn in 8 octaven, is het verschil tussen N52 en N53 bijvoorbeeld een halve toon.
In feite worden de 8 octaven, die beschikbaar zijn in het N-commando, een halve toon opgeschoven ten opzichte van het O-commando. Dit komt omdat N0, hetgeen zou moeten corresponderen met de C in O1, een stilte is die derhalve als rust kan worden gebruikt.
De hoogste toon met het O-commando is B in O8. N96 echter, ligt een halve toon hoger en speelt de naastgelegen hogere C.
Let er op dat de middelste C gelijk is aan N36.



Het volgende programma speelt dezelfde E-majeur, maar ditmaal met het N-commando:

```
10 REM E-majeur: N-notatie
20 PLAY "N40N42N44N45N47N49N51"
30 PLAY "N52N54N56N57N59N61N63N64"
```

Als u muziek kunt lezen, vindt u de A-G-notatie ongetwijfeld gemakkelijker te gebruiken.
Tot nu toe heeft u alleen noten gespeeld met dezelfde duur of lengte. Om de lengte van een noot te wijzigen kunt u het commando L gebruiken. Dit commando moet u laten volgen door een getal dat mag liggen tussen 1 en 64. De meest nuttige waarden volgen hieronder:

Ln	duur van de noot
L1	speelt een hele noot
L2	speelt een halve noot
L4	speelt een kwartnoot
L8	speelt een achtste noot

L16	speelt een zestiende noot
L32	speelt een tweeëndertigste noot

U kunt ook L3 gebruiken enz. L3 zou een derde van een hele noot produceren en kan worden gebruikt voor het spelen van een triool.
Als u geen lengte meegeeft voor uw noten, worden deze automatisch als een kwartnoot gespeeld, dus L4 is de beginwaarde of vervangingswaarde.
Om de schaal van E-majeur te wijzigen, opdat deze wordt gespeeld in achtste noten, dient u L8 toe te voegen aan het begin van de PLAY-string. Het L-commando beïnvloedt alle navolgende noten in een programma.
Het volgende programma speelt E-majeur in achtste noten, met behulp van de A-G-notatie en daarna de N-notatie. Let er op dat de toonlengte slechts eenmaal in het programma hoeft te worden opgegeven:

```
10 REM E-majeur: achtsten
20 PLAY "L8EF+G+ABO5C+D+E"
30 PLAY "N40N42N44N45N47N49N51N52"
```

Om de lengte in te stellen van een enkele toon, mag u de letter L weglaten en het getal opgeven dat de lengte aangeeft van de noot, na de te spelen noot:

`A+16` is gelijk aan `L16A+`

Let er op dat u dit niet kunt doen bij de N-notatie.
Om een rust op te geven kunt u het commando R gebruiken. De vervangingswaarde is hier R4, wat overeenkomt met een kwart rust. Om de lengte van de rust te veranderen moet u R laten volgen door een getal tussen 1 en 64. Dit getal geeft de duur aan van de rust. Dus:

Rn	duur van de rust
R1	een hele rust
R2	een halve rust
R3	een derde rust
R4	een kwart rust
R8	een achtste rust
R16	een zestiende rust
R32	een tweeëndertigste rust
R64	een vierenzestigste rust

Als u een noot met een stip (door de stip wordt de halve waarde van de noot toegevoegd) of een rust wilt spelen, laat dan dat commando volgen door een stip of een volle stop. Een stip na een noot resulteert in een verlenging van de duur van de toon met de helft van de oorspronkelijke lengte. Bijvoorbeeld:

```
10 REM noten met stip
20 PLAY "AR64A.R64A.."
```

Dit speelt toon A als een vierde, gevolgd door een vierde noot met een stip en ten slotte een vierde noot met twee stippen.
De rusten zijn opgenomen opdat u de tonen individueel kunt horen. Zonder deze rusten zou U toon A één keer horen spelen.
Het tempo van een muziekstuk kan worden bepaald met behulp van het T-commando. Het getal dat volgt op dit commando kan iedere waarde aannemen tussen 32 en 255 en bepaalt het aantal kwartnoten dat wordt gespeeld in een minuut. U kunt daarom het tempo van een muziekstuk laten variëren van een langzame 32 kwartnoten per minuut, tot een snelle 255!
De vervangingswaarde voor het tempo is T120.
In het laatste voorbeeld van de E-majeur-schaal hebben we alle noten in achtste noten veranderd, zodat de schaal twee keer sneller wordt gespeeld dan in het vorige voorbeeld. Een andere manier om de schaal te versnellen zou kunnen worden bereikt door het tempo op T240 te zetten. De noten worden dan nog steeds als achtste noten gespeeld, maar iedere achtste noot wordt twee keer sneller gespeeld dan ervoor:

```
10 REM Majeur: Sneller Tempo
20 PLAY "T240EF+G+ABO5C+D+E"
30 PLAY "N40N42N44N45N47N49N51N52"
```

Het tempo hoeft maar één keer te worden gezet in het programma.
Het commando V zet het volume. Het getal dat op V volgt moet liggen tussen 0 en 15. V0 is zeer zacht of pianissimo; V15 is erg hard of fortissimo.
De vervangingswaarde is V8.
Evanals bij de andere commando's, O, L en T, geldt dat zodra de waarde is gezet, deze gedurende het gehele programma actief is, totdat u deze wijzigt of de computer uitschakelt. Als een frase vaak wordt herhaald in een muziekstuk, kunt u de string die de frase bevat toewijzen aan een string-variabele. Indien u de stringvariabele in PLAY gebruikt, moet u de naam van de variabele laten voorafgaan door het X-commando en deze laten volgen door een puntkomma.
Twee stringvariabelen, A$ en B$ worden in het volgende voorbeeldprogramma gebruikt. Het tempo wordt ingevoerd en opgeslagen in de numerieke variabele TEMPO, opdat u het effect kunt zien van het wijzigen van deze waarde. Probeer TEMPO-waarden van 120 en 240:

```
10 REM DRINK TO ME ONLY:MELODIE
20 INPUT "TEMPO";TEMPO
30 A$="V8AR64AR64AB-2R64B-O5CO4B-AGA"
40 B$="B-O5CO4FB-A2GF2.F."
50 GOSUB 1000
60 GOSUB 1000
70 PLAY "V9O5CR64CO4AO5CF2CR64CO4AO5CR64C2"
80 PLAY "CD2CR64CO4B-AR64A2.G2."
90 GOSUB 1000
100 END
990 REM SUBROUTINE
1000 PLAY"T=TEMPO;XA$;"
```

```
1010 PLAY"XB$;"
1020 RETURN
```

Als u dit programma intikt, moet u uitkijken voor het intikken van de letter O, voor de octaafcommando's, en niet het getal 0. Houd ook rekening met stippen en puntkomma's.
Let er op dat de lengte van de tonen individueel wordt bepaald.
De 'R64's worden gebruikt om de geluiden te scheiden, als eenzelfde noot twee keer wordt gespeeld.
Bij het overnemen van dit geluid is iedere muziekfrase in een PLAY-opdracht geplaatst, om de controle op tikfouten te vereenvoudigen. Dit kan echter leiden tot enorm lange programmaregels; vergeet niet dat u tot maximaal 255 tekens kunt hebben in een programmaregel.
De MSX-computer kan drie geluiden tegelijkertijd produceren. We kunnen daarom nog twee geluiden toevoegen aan ons muziekstuk. Het volgende programma laat zien hoe:

```
10 REM DRINK TO ME ONLY: 3 DELEN HARMONIE
20 INPUT "TEMPO";TEMPO
30 A1$="V9AR64AR64AB-2R64B-O5CO4B-AGA"
40 A2$="V7FEDCDEFEFEF"
50 A3$="V7O3F2.G2.AGFB-A"
60 B1$="B-O5CO4FB-A2GF2.F2."
70 B2$="EC2DRCO3B-RO4CDCDC"
80 B3$="GAO2BO3C2.F2."
90 GOSUB 1000
100 GOSUB 1000
110 PLAY"V10O5CR64CO4AO5CF2CR64CO4AO5CR64C2","V8EF2GF2G
F2EGF","V8B-F2ED2EF2GA2"
120 PLAY"CD2CR64CO4B-AR64A2.G2.","E-DEFR64F2R64FR64F2.E
2.","GB-2AGO2GAB2.O3C2."
130 GOSUB 1000
140 END
990 REM SUBROUTINE
1000 PLAY"T=TEMPO;XA1$;","T=TEMPO;XA2$;","T=TEMPO;XA3$;
"
1010 PLAY "XB1$;","XB2$;","XB3$;"
1020 RETURN
```

De strings die bij de drie geluiden horen worden opgesomd in elk van de PLAY-opdrachten. De strings dienen van elkaar te worden gescheiden met behulp van komma's. Let er op dat tempo, volume enz. onafhankelijk voor ieder geluid worden bepaald.
Geluid 1 speelt steeds met een iets hoger volume. Dit is omdat dit geluid de melodie speelt.
U heeft wellicht opgemerkt dat bij het einde van iedere frase de geluiden iets uit de pas gaan lopen. Dit komt omdat de strings voor ieder geluid verschillende commando's bevatten; sommige strings hebben heel wat meer rusten en octaafveranderingen dan andere. Ieder commando kost een zeer

kleine, beperkte uitvoeringstijd. U kunt proberen de geluiden te synchroniseren door een aantal 'R64' en octaafwijzigingen op geschikte plaatsen toe te voegen.
(Opmerking: Een O4-commando voert niets uit, als de string zich reeds in deze octaaf bevindt, maar verhoogt toch de uitvoeringstijd van de string.)
Tot nu toe klonken alle tonen gelijkmatig. Het is mogelijk dit te wijzigen, door een soort van envelope boven op de noot te plaatsen; deze envelope bestuurt de kwaliteit van de noot. Er bestaan verschillende vooraf gedefinieerde envelope-configuraties waaruit u een keuze kunt maken. Zie hiervoor Sectie 3: GELUID.
Om een envelope te selecteren, gebruikt u het S-commando, gevolgd door een getal tussen 0 en 15. De omlooptijd van de envelope of modulatie, wordt bepaald door het M-commando. Deze kan worden gezet op elke waarde tussen 1 en 65535; de vervangingswaarde is 255. Tik het volgende in:

```
10 PLAY "S10L1CDE"
```

U hoort de tiende envelope die boven op de drie noten C, D en E is gelegd. Probeer een envelopenummer te wijzigen. Zodra u denkt de idee hiervan te hebben begrepen kunt u de modulatie wijzigen met behulp van het M-commando. Bijvoorbeeld:

```
10 PLAY "M1000S10L1CDE"
```

Probeer andere waarden van M, bijvoorbeeld M10, M6000 enz. met dezelfde envelope.
Experimenteer met andere waarden voor M en S. De volgorde van M en S is onbelangrijk, maar zij dienen te komen voordat de noten worden gespeeld.
Ten slotte het BEEP-commando. Dit is de eenvoudigste manier om de computer geluid te laten maken. Tik in:

```
BEEP <RETURN>
```

Na het uitvoeren van BEEP zet de computer alle geluidsvariabelen terug op de vervangingswaarden. Dus alle muziek-macrocommando's staan op de beginwaarde.

Deel 2
Geavanceerde programmeergids

Hoofdstuk 1

Geavanceerde programmabewerkingen

MSX-BASIC beschikt over een volledige schermbewerkingsmogelijkheid, die u in staat stelt om een programma op iedere plaats van het scherm te bewerken.
Dit hoofdstuk gaat in op de technische details van deze bewerkingen en op de geavanceerde bewerkingsmogelijkheden van MSX met behulp van de besturingstoets <CTRL>.

Hoe een programma wordt bewerkt

Gedurende het schrijven van een programma is het dikwijls nodig om een serie regels te schrijven of om hele stukken van uw programma te schrappen. MSX-BASIC heeft vier zeer handige commando's die het bewerkingsproces aanzienlijk vergemakkelijken: LIST, AUTO, RENUM en DELETE.

LIST

Zodra u een programma heeft ingebracht in het geheugen van de computer, wilt u wellicht eerst even alles nakijken voordat u het programma laat uitvoeren. Om het programma van boven naar beneden te bekijken kunt u LIST gebruiken. Dit commando toont het programma in een duidelijke logische volgorde; u kunt nu het programma bewerken met behulp van de verschillende bewerkingsmethoden van MSX.
De LIST-opdracht kent de volgende variaties, die in uw behoeften kunnen voorzien:

LIST	toont het volledige programma
LIST <REGEL>	toont een specifieke regel
LIST <REGEL1><REGEL2>	toont <REGEL1> tot en met <REGEL2>
LIST <REGEL>-	toont het programma vanaf <REGEL>
LIST -<REGEL>	toont het programma tot aan <REGEL>

U zult zien dat als het programma redelijk lang is, LIST het programma laat doorbewegen op het scherm, totdat de laatste regel, of de in het LIST-commando opgegeven regel wordt bereikt.

Er zijn twee manieren om een relevant gedeelte van het programma te tonen:

A. Gebruik LIST <REGEL1>-<REGEL2> opdat u alleen dat gedeelte te zien krijgt dat u zien wilt.
B. Gebruik LIST en wacht tot u het gedeelte te zien krijgt dat u wenst; druk dan op <CTRL> <STOP> om uit de programmalijst te komen.

Let er op dat de <STOP>-toets slechts tijdelijk de programmalijst stopt. Drukt u nogmaals op <STOP> dan wordt de lijst voortgezet, vanaf de plaats waar werd gestopt.

AUTO

Het AUTO-commando plaatst u in de automatische regelnummeringmode, die vanzelf een regelnummer aanbiedt, nadat u op <RETURN> heeft gedrukt. Het bespaart u de moeite van het intikken van het regelnummer voor iedere regel en vergemakkelijkt het leven van de programmeur. De regelnummers hebben een constante verhogingsfactor.
Het AUTO-commando kent de volgende mogelijkheden, die in uw behoeften kunnen voorzien:

AUTO	geeft regelnummers vanaf 10 met een verhogingsfactor 10.
AUTO <REGEL>	geeft regelnummers vanaf <REGEL> met een verhogingsfactor 10.
AUTO <REGEL>,<FACTOR>	geeft regelnummers vanaf <REGEL> met een verhogingsfactor <FACTOR>.
AUTO ,<FACTOR>	start vanaf de regel die u zojuist verliet met een verhogingsfactor <FACTOR>.
AUTO <REGEL>	geeft regelnummers vanaf <REGEL> met dezelfde verhogingsfactor als ervoor werd gebruikt.

Twee van de meest gangbare toepassingen van AUTO tijdens het schrijven of bewerken van een programma zijn:

1. Als u een programma invoert dat u reeds op papier heeft geschreven, gebruikt u AUTO om regelnummers vanaf 10 met een verhoging van 10 te verkrijgen. Dit levert een redelijk geordend programma op.
2. Gesteld, u heeft de volgende regelnummers:
   100, 110, 120, 130, 140, 150, en 160 en u wilt een subroutine toevoegen tussen de regels 150 en 160. Als eerste kunt u dan ruimte creëren met behulp van RENUM 1000,150,100. Het resultaat levert de volgende regels op:
   100, 110, 120, 130, 140, 1000, 1100. Gebruik vervolgens AUTO 1010,5 hetgeen u een serie regelnummers aanbiedt, beginnend bij 1010 en

met een verhoging van 5. Het uiteindelijke resultaat ziet er als volgt uit:
100, 110, 120, 130, 140, 1000, <1010, 1015, 1020, 1025, ...>, 1100.

Er zijn enkele punten waaraan u moet denken bij AUTO:

1. U kunt op twee manieren uit de AUTO-mode komen:
   a. <CTRL> <STOP> tegelijkertijd indrukken;
   b. <CTRL> <C> tegelijkertijd indrukken.
   De regel van waaruit u onderbrak wordt niet bewaard.
2. Als u een regel bewerkt die reeds bestaat, geeft de computer u een signaal, '*' achter het regelnummer. Als u de regel niet wilt overschrijven, druk dan op <RETURN>. Wilt u de regel juist vervangen, negeer de '*' dan en tik hetgeen in dat u van plan was.

RENUM

Als u een programma heeft ingevoerd, blijkt soms dat de regelnummers geen constante verhoging hebben. Dit lijkt erg slordig, en als het programma moet worden getoond aan derden geeft het wellicht een slechte indruk. De remedie is eenvoudig: hernummer het hele programma met behulp van RENUM. RENUM op zichzelf, hernummert vanaf regel 10 met een verhoging van telkens 10. Dit is de meest gangbare vorm, maar u kunt vanaf iedere willekeurige regel met hernummeren beginnen, met verschillende verhogingen. RENUM past automatisch de regelnummers aan die worden gerefereerd vanuit GOTO, GOSUB, RESTORE, THEN, ON GOTO en ON GOSUB.
RENUM kent de volgende mogelijkheden, die in uw behoeften kunnen voorzien:

RENUM	hernummert vanaf regel 10 met een verhoging van 10.
RENUM <REGEL1>	hernummert vanaf de nieuwe <REGEL1> met een verhoging van 10.
RENUM <REGEL1>,<REGEL2>	hernummert vanaf de oude <REGEL2>, te beginnen met <REGEL1> en een verhoging van 10.
RENUM <REGEL1>,<REGEL2>, <FACTOR>	hernummert vanaf de oude <REGEL2> te beginnen met <REGEL1> en een verhoging <FACTOR>.
RENUM ,<REGEL2>,<FACTOR>	hernummert vanaf de oude <REGEL2> vanaf regel 10 met een verhoging <FACTOR>.
RENUM ,<REGEL2>	hernummert vanaf de oude <REGEL2>, vanaf regel 10 met een verhoging 10.
RENUM ,,<FACTOR>	hernummert vanaf regel 10 met een verhoging <FACTOR>.
RENUM <REGEL1>,,<FACTOR>	hernummert vanaf de nieuwe <REGEL1> vanaf de oude regel 10 met een verhoging <FACTOR>.

RENUM wordt ook gebruikt om voldoende regelruimte te creëren, zodat u een gedeelte in een programma kunt toevoegen (zoals werd gedaan bij het voorbeeld met AUTO, hiervoor).

DELETE

Soms lijkt een bepaald gedeelte van het programma een puinhoop. Als het gebied in het programma dat u wilt verwijderen slechts één regel betreft, is het invoeren van alleen het regelnummer voldoende om dit te bereiken. Wilt u echter een hele serie regels tegelijk verwijderen, gebruik dan het DELETE-commando. Dit is eenvoudig, effectief en ook permanent, dus als u uw regels kwijt bent, krijgt u ze niet meer terug, anders dan door ze opnieuw in te tikken.
Het DELETE-commando kent de volgende variaties, die in uw behoeften kunnen voorzien:

DELETE <REGEL1>	schrapt <REGEL1>.
DELETE <REGEL1>-<REGEL2>	schrapt <REGEL1> tot en met <REGEL2>.
DELETE -<REGEL2>	schrapt alles tot en met <REGEL2>.

Besturingstoetsen en speciale functietoetsen

MSX heeft een aantal speciale functietoetsen en besturingstoetsen die hoofdzakelijk voor bewerking worden gebruikt. Hier volgt een tabel die aangeeft wat ze doen en hoe u ze kunt benaderen. Alle besturingstoetsen verkrijgt u via de <CTRL>-toets en de relevante toets, door deze tegelijkertijd in te drukken:

HEX	BESTURINGS-TOETS	SPECIALE TOETS	FUNCTIE
02	* <CTRL><B>		beweegt de cursor één woord naar links.
03	* <CTRL><C>		stopt de uitvoering, als BASIC op INPUT wacht. Keert terug naar de commando-mode.
05	* <CTRL><E>		schrapt alles wat zich rechts van de cursor bevindt, inclusief het teken onder de cursor.
06	* <CTRL><F>		beweegt de cursor een woord naar rechts.
07	* <CTRL><G>		BEEP.
08	<CTRL><H>	BS	gaat een spatie terug en schrapt daarbij het teken links van de cursor, waarbij alles rechts van de cursor een positie naar links opschuift.
09	<CTRL><I>	TAB	TAB verplaatst naar de eerstvolgende

			TAB-positie. TAB wordt gezet op intervallen van 8 tekens. Er komen spaties op de plaatsen van waaruit wordt verplaatst.
0A	* <CTRL><J>		volgende regel. Beweegt de cursor naar het begin van de volgende regel.
0B	* <CTRL><K>	HOME	verplaatst de cursor naar zijn uitgangspositie linksboven op het scherm.
0C	* <CTRL><L>	CLS	wist het scherm en verplaatst de cursor naar zijn uitgangspositie linksboven op het scherm.
0D	* <CTRL><M>	RETURN	wagen terug. Voert de huidige regel in.
0E	* <CTRL><N>		verplaatst de cursor naar het einde van de huidige regel.
12	* <CTRL><R>	INS	gaat over op INSERT. De cursor wordt half zo groot en stelt u in staat om tekens in te voegen op de plaats van de cursor, zonder daarbij iets te schrappen.
15	* <CTRL><U>		schrapt huidige regel.
18	* <CTRL><X>	SELECT	genegeerd door huidige MSX-versie.
1B	* <CTRL><<>	ESC	genegeerd door huidige MSX-versie.
1C	* <CTRL><Y>	→	cursor naar rechts.
1D	* <CTRL><>>	←	cursor naar links.
1E	* <CTRL><^>	↑	cursor omhoog.
1F	* <CTRL><_>	↓	cursor omlaag.
7F	* <CTRL><DEL>	DEL	schrapt teken onder de cursor en verplaatst alles rechts van de cursor een positie naar links.

* Geeft aan dat de INSERT-mode wordt gedeactiveerd, indien deze actief was.

Hoofdstuk 2

Constanten en variabelen

Constanten

Constanten zijn getallen of strings die niet variëren, bijvoorbeeld 100 en 'ACTIE' zijn constanten.
Er bestaan twee soorten constanten, strings en getallen.

Stringconstanten

Een stringconstante is een reeks van tekens tot 255 tekens lang. Hij kan bestaan uit elke soort tekens, grafische tekens of besturingscodes, die worden gebruikt in de standaard-MSX-BASIC. Ze moeten worden ingesloten door aanhalingstekens, dus:

```
"HEER BOMMEL"
"*--WAT IS DIT?"
"ARIE, ROGIER EN GIJS"
"FL.100.000.000"
```

Numerieke constanten

Er bestaan negatieve en positieve constanten, in zes verschillende vormen.

1. Integere constanten	Gehele getallen tussen -32768 en 32767 of 1111111111111111 tot 0000000000000000 in 16 bits. Ze bevatten geen decimale punt.
2. Vaste punt constanten	Positieve of negatieve reële getallen, die soms een decimale punt bevatten bijv. 657.188.
3. Drijvende punt constanten	Positieve of negatieve reële getallen in exponentiële vorm. Formaten: enkelvoudige nauwkeurigheid <integer>E<integer> -56E10 <vaste-punt>E<integer> 7.865E5.

	dubbele nauwkeurigheid <integer>B<integer> -18962432662D50 <vaste-punt>D<integer> 7.8827726265D-5. De nauwkeurigheidsmarge ligt tussen 1E-64 tot 1E64.
4. Hexadecimale getallen	Hexadecimale getallen worden voorzien van het voorvoegsel &H. Hex is een getallenstelsel met grondtal 16 en gebruikt 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Voorbeelden: &HFF (1 byte lang) &H8D1A (2 bytes lang)
5. Octale getallen	Octale getallen krijgen het voorvoegsel &0. Het binaire getallenstelsel heeft het grondtal 8, dus gebruikt 0, 1, 2, 3, 4, 5, 6, 7. Voorbeeld: &06543.
6. Binaire getallen	Binaire getallen krijgen de prefix &B. Het binaire getallenstelsel heeft het grondtal 2, dus gebruikt alleen 1 en 0. Voorbeeld: &B01010101.

Enkelvoudige nauwkeurigheid en dubbele nauwkeurigheid

Eén van de voornaamste eigenschappen van MSX-BASIC is dat het in dubbel nauwkeurige BCD-rekenfuncties beschikt over een nauwkeurigheid van 14 cijfers. Dit is een nauwkeurigheid waarover de meeste 8-bit machines niet beschikken. Dubbele nauwkeurigheid wordt meestal alleen bij 16- en 32-bit systemen geleverd, maar Microsoft heeft zijn 4.5 MS-BASIC herschreven voor het toepassen van nauwkeurige berekeningen. Er zijn zowel enkelvoudige als dubbele nauwkeurige numerieke constanten mogelijk, maar als u dit niet aangeeft veronderstelt de computer dubbele nauwkeurigheid.
Nauwkeurigheid:

enkelvoudige nauwkeurigheid	- 6 cijfers
dubbele nauwkeurigheid	- 14 cijfers

Enkelvoudig nauwkeurige constanten hebben de volgende eigenschappen:

1. Exponentiële vorm E 1.65E-2
2. Meegegeven uitroepteken 235.77!

Dubbel nauwkeurige constanten hebben de volgende eigenschappen:

1. Ieder getal of cijfer zonder een uitroepteken of exponent.
   1878932.2 156

2. Exponent met behulp van D. 562.765333D-6
3. Meegegeven #-teken. 387.001#

Variabelen

Variabelen zijn namen waaraan u een zekere waarde kunt toekennen. Deze waarden kunt u veranderen. Zij kunnen specifiek door de programmeur worden toegekend (bijv. PI=3.14) of worden berekend door de computer (bijv. PI=4*ATN(1)).

Namen van variabelen

De namen van variabelen kunnen iedere lengte aannemen, de computer gebruikt echter alleen de eerste twee letters van de namen van variabelen. Als twee variabelen de eerste twee letters gelijk hebben, worden zij verondersteld gelijk te zijn, dus VAR1 en VAR2 zijn gelijk. Wilt u ze laten verschillen, dan moet u er V1 en V2 van maken.
De namen van variabelen mogen geen gereserveerde woorden bevatten, dus geen BASIC-sleutelwoorden. Bijvoorbeeld LENGTE% is fout, omdat dit LEN bevat. Als een variabele begint met FN, wordt dit verondersteld een zelf gedefinieerde functie te zijn.

Declaratie van tekensoort

Als een naam van een variabele niet is voorzien van een type-declaratie, wordt er uitgegaan van een dubbel nauwkeurige numerieke variabele.

A1=<dubbel nauwkeurig getal>

Toegevoegde speciale tekens:

- $ Een dollarteken geeft aan dat de variabele een string-variabele is. Z$="string-tekens", ADRES$="KALVERSTRAAT 25".
- % Een procentteken geeft een integere variabele aan. X%=25, ZXY%= -32768.
- ! Een uitroepteken geeft een enkelvoudig nauwkeurige numerieke variabele aan. F!=5679.7!, SPRES!=4.2888e-02.
- # Dit geeft een dubbel nauwkeurige numerieke variabele aan. H#=3.14161718293, MAXIMUM#=5277.76525D25.

DEF <tekensoort>

Het is mogelijk om een tekensoort te definiëren met behulp van DEF-opdrachten. Als het soort variabelen wordt gedeclareerd met behulp van deze opdrachten, wordt iedere variabele die begint met het gedeclareerde teken,

gedeclareerd in deze tekensoort zonder dit te declareren in de naam van de variabele. Dit betekent dat als u DEFSTR A definieert, hierna iedere variabele die begint met A een string-variabele wordt, zonder dat daarbij gebruik wordt gemaakt van het dollarteken. Echter, dit houdt niet in dat AB% ook een string-variabele wordt, omdat de declaratie van de integer (dus %) prioriteit heeft boven DEFSTR.
Er zijn vier DEF <tekensoort>-opdrachten. Deze luiden als volgt:

DEFSTR-opdracht — declareert dat iedere naam van een variabele, die begint met de opgegeven reeks tekens, moet worden behandeld als een string-variabele:

```
DEFSTR A
AST="PLANEET"
```

DEFINT-opdracht — geeft aan dat iedere naam van een variabele die begint met de opgegeven reeks tekens, moet worden behandeld als een integer getal.

DEFSNG-opdracht — geeft aan dat iedere naam van een variabele die begint met de opgegeven reeks tekens, moet worden behandeld als een enkelvoudig nauwkeurig getal.

DEFDBL-opdracht — geeft aan dat iedere naam van een variabele die begint met de opgegeven reeks tekens, moet worden behandeld als een dubbel nauwkeurig getal.

Let er op dat de soort-verklaringen van de symbolen #, $, % en ! prioriteit hebben boven de genoemde opdrachten.

Array (DIM)

Een array is een groep variabelen met een gemeenschappelijke naam. Deze wordt opgezet met behulp van de DIM-opdracht. DIM-opdrachten reserveren een gedeelte van het geheugen voor het gebruik van de array, voordat deze wordt gebruikt. Ieder element van de array heeft een label. Stel bijvoorbeeld dat u de array DIM C(3) heeft opgezet. U beschikt dan over vier variabelen met de naam C(), bijvoorbeeld C(0), C(1), C(2) en C(3). Alle arrays beginnen met een nul-element. Het betreft hier een één-dimensionale array, doch dit is geen reden om geen multi-dimensionale arrays te hebben, zoals DIM A(2,2) die een tabel oplevert van 3 bij 3 variabelen zoals hieronder:

```
A(0,0)  A(1,0)  A(2,0)
A(0,1)  A(1,1)  A(2,1)
A(0,2)  A(1,2)  A(2,2)
```

Dit is een twee-dimensionale tabel, maar u kunt ook gebruik maken van een drie-dimensionale array of een tien-dimensionale. De maximum-dimensie die mogelijk is, is 255, maar u zult in dit laatste geval een tekort aan geheugen hebben om een dergelijke array op te slaan.
Als een array over minder dan 12 elementen beschikt, bijvoorbeeld DIM A(10), dan hoeft deze DIM niet te worden uitgevoerd, omdat de computer automatisch 11 geheugenposities reserveert. Programma's zoals hieronder zijn volkomen legaal, zelfs zonder een DIM-opdracht.

```
10 FOR I=0 TO 10
20 S(I)=1
30 NEXT I
```

Het is mogelijk om arrays van iedere tekensoort te hebben, zolang de soort aangegeven is. Bijvoorbeeld, DIM S$(100) levert u honderd en één elementen voor een string-array.
Als de array wordt geïnitialiseerd, worden bij numerieke arrays alle waarden verondersteld gelijk aan nul te zijn en bij string-arrays worden nul-strings verondersteld.

Geheugenbenodigdheden

Variabelen worden opgeslagen in een DIM-gebied, een VARIABELEN-gebied of een STRING-gebied, afhankelijk van de soort (zie de geheugenorganisatie).
Hier volgt een lijst van de hoeveelheid bytes die door deze variabelen wordt gebruikt:

variabelen	integer	2 bytes
	enkelvoudige nauwkeurigheid	4 bytes
	dubbele nauwkeurigheid	8 bytes
	string	3 bytes plus lengte van de inhoud per element
arrays	integer	2 bytes per element
	enkelvoudige nauwkeurigheid	4 bytes per element
	dubbele nauwkeurigheid	8 bytes per element
	string	3 bytes plus lengte van de inhoud per element

VARPTR

Om te bepalen waar de gegevens van een variabele zijn opgeslagen in het geheugen, kunt u een speciale functie gebruiken, namelijk VARPTR die u het adres geeft van een specifieke variabele, bijvoorbeeld:

```
10 A%=100
20 PRINT VARPTR(A%)
```

U kunt zo de geheugenpositie van iedere soort variabele bepalen, zolang

als deze bestaat. Dit kan een string zijn of zelfs een array.

String-variabelen-gebied

Alle inhoud van string-variabelen wordt opgeslagen in een speciaal STRING-geheugengebied (zie de geheugenorganisatie). De grootte van dit gebied is beperkt tot een standaardwaarde van 200 posities; daarom is de maximale hoeveelheid van een string op ieder moment beperkt tot 200 tekens. U kunt echter deze grootte wijzigen met behulp van de CLEAR-opdracht. Om bijvoorbeeld de grootte van het STRING-gebied te verhogen tot 255 bytes, moet u de computer CLEAR 255 laten uitvoeren.

Hoofdstuk 3

Soortverandering

MSX-BASIC kan het ene soort numerieke constante wijzigen in een andere. Hiervoor bestaat een aantal regels:

1. Als een numerieke constante van het ene soort gelijk wordt gesteld aan een andere soort, hangt het resultaat volledig af van het soort variabele.

Voorbeeld 1

```
10 C!=1.2345678901
20 PRINT C!
RUN
1.23456
```

Het dubbel nauwkeurige getal 1.2345678901 wordt gewijzigd in enkelvoudige nauwkeurigheid als dit wordt gelijkgesteld aan C!, een enkelvoudig numerieke variabele.

Voorbeeld 2

```
10 I%=1.23E-03
PRINT I%
RUN
0
```

Het enkelvoudig nauwkeurige getal 1.23E-03 wordt afgerond tot de dichtstbij gelegen integere waarde, die nul is indien deze wordt gelijkgesteld aan een integere variabele.

2. Gedurende de evaluatie van de uitdrukking worden alle variabelen en constanten aangepast aan de meest geschikte nauwkeurigheid van de gebruikte variabelen.
   Als de uitdrukking wordt gelijkgesteld aan een variabele, wordt de nauwkeurigheid aangepast aan de nauwkeurigheid die voor de variabele van toepassing was. Als het soort van de variabele niet is opgegeven, dan blijft het resultaat in dezelfde naukeurigheid waarin de evaluatie plaatsvond.

Voorbeeld 1

```
10 D=1!/3#
20 PRINT D
RUN
0.3333333333333
```

1! is in enkelvoudige nauwkeurigheid, terwijl 3# dubbel is. De berekening werd uitgevoerd in dubbele nauwkeurigheid en het resultaat werd opgeslagen in de variabele D met dubbele nauwkeurigheid.

Voorbeeld 2

```
10 D=1!/3!
20 PRINT D
RUN
.333333
```

Beide constanten staan in enkelvoudige nauwkeurigheid, dus de berekening werd uitgevoerd in enkelvoudige nauwkeurigheid. Omdat D een variabele met dubbele nauwkeurigheid is, wordt het resultaat hierin aangepast, doch mist de accuratesse van dubbele nauwkeurigheid.

Voorbeeld 3

```
10 D!=1/3
20 PRINT D!
RUN
.33333
```

De berekening werd uitgevoerd in dubbele nauwkeurigheid, doch het resultaat wordt aangepast in enkelvoudige nauwkeurigheid, als dit wordt gelijkgesteld aan D!. Het resultaat wordt afgekort.

3. Logische bewerkers wijzigen hun constanten en variabelen in 16-bit integere waarden, voordat de bewerking wordt uitgevoerd. Het resultaat is een integere waarde en de betrokken getallen moeten voldoen aan het integere gebied, omdat anders een 'Overflow Error' optreedt.

Voorbeeld

```
10 D%=156.65 AND 654
20 PRINT D%
RUN
140
```

De getallen worden gewijzigd in 156 en 654. Het resultaat 140 wordt opgeslagen in D%. Dit is een integere variabele.

4. Indien een waarde met een drijvende of vaste punt wordt gewijzigd in een integere waarde, wordt het gebroken gedeelte afgekort.

Voorbeeld 1

`10 A%=123.456`
`20 PRINT A%`
`RUN`
`123`
	De waarde 123.456 met vaste punt wordt afgekort tot 123 indien hij wordt toegekend aan de integere variabele A%.

Voorbeeld 2

`10 A%=1.0097554D02`
`20 PRINT A%`
`RUN`
`100`
	Het getal 1.0097554D02 met drijvende punt wordt afgerond op 100, indien dit wordt gelijkgesteld aan de integere variabele A%.

5. Indien een enkelvoudig nauwkeurig getal wordt gewijzigd in een dubbel nauwkeurig getal, kunnen alleen maar 6 cijfers worden meegegeven in het dubbel nauwkeurige resultaat.

Voorbeeld

`10 A#=1.33333`
`20 PRINT A#`
`RUN`
`1.33333`
	A# is een dubbel nauwkeurige variabele, maar dit maakt het aan A# toegekende getal daarom niet nauwkeuriger - het resultaat blijft 1.33333.

Soort aanpassing met behulp van MSX-BASIC-functies

CDBL, CINT, CSNG, VAR, STR$

Ondanks het feit dat de MSX automatische soortaanpassing doet gedurende het evalueren van uitdrukkingen, kunt u de soortaanpassing inprogrammeren, en indien nodig, kunt u daarbij gebruik maken van één van de aanpassingsfuncties. CDBL, CINT en CSNG volgen dezelfde regels als eerder vermeld, maar VAR en STR$ zijn achtereenvolgens string naar numeriek en numeriek naar string en volgen iets andere regels.

CINT

CINT wijzigt enkelvoudige en dubbel nauwkeurige getallen, variabelen en uitdrukkingen in integere waarden. Het argument moet liggen in het gebied -32768 tot 32767. Is dit niet het geval dan treedt een fout op, 'Overflow Error'.

CINT(<num-const>)
CINT(<num-var>)

CINT(<num-exp>)

Voorbeeld

```
PRINT CINT(1234.56789)
1234
```

CSNG

CSNG is een functie die het argument wijzigt in enkelvoudige nauwkeurigheid. CSNG rondt af tot op de dichtstbijgelegen zesde decimale positie. Indien een integere waarde wordt aangepast, blijft de accuratesse intact.

CSNG(<num-const>)
CSNG(<num-var>)
CSNG(<num-exp>)

Voorbeeld

```
PRINT COS(0.7656)
.72096670541357
PRINT CSNG(COS(0.7656))
.720967
```

CDBL

De CDBL-functie wijzigt integere en enkelvoudig nauwkeurige getallen, variabelen en uitdrukkingen in dubbel nauwkeurige getallen. Indien een enkelvoudig nauwkeurig getal wordt aangepast, worden alleen de eerste zes cijfers meegegeven aan het dubbel nauwkeurige resultaat.

Voorbeeld

```
PRINT CDBL (5.666!)
5.666
```

VAL en STR$

Deze functies worden gebruikt om strings naar getallen en getallen naar strings om te zetten.

STR$(<getal>)

STR$ wijzigt het numerieke argument in een string.

VAL(<string>)

VAL geeft de waarde van een string die een getal bevat. Als een string een string-voorstelling is van een getal, kan VAL worden toegepast. Echter, een uitdrukking met een string werkt niet. VAL haalt de spaties uit de string en ook de tabs en volgende regel-tekens die vooraf kunnen gaan aan een getal in een string; andere tekens kan VAL niet aan. VAL reageert wel op een plus- of minteken.
Zie voor voorbeelden ook sectie 1, hoofdstuk 15.

Hoofdstuk 4

Uitdrukkingen en bewerkers

Een uitdrukking kan een string zijn, een numerieke constante of iedere combinatie hiervan die een enkelvoudige waarde oplevert, zolang deze acceptabel is. Er zijn vier soorten logische of wiskundige bewerkers:

1. Rekenkundig
2. Relationeel
3. Boolean (logisch)
4. Functioneel

1 en 2 worden in dit hoofdstuk behandeld. We verwijzen verder naar de desbetreffende secties voor de andere twee bewerkers.

Rekenkundige bewerkers

De rekenkundige bewerkers zijn de volgende, op volgorde van prioriteit:

1.	^ exponentieel	A^B
2.	- ontkenning	-A
3.	*, / vermenigvuldiging	A*B, A/B
4.	+, - optelling en aftrekking	A+B, A-B

DIV en MOD

DIV wordt in de MSX-BASIC voorgesteld door ¥ (yen). DIV voert een integere deling uit. Modulus wordt voorgesteld door MOD.
Wat zijn DIV en MOD? Het volgende voorbeeld laat zien hoe ze worden gebruikt. In de normale deling levert 10 gedeeld door 4, 2.5 op. Evenzo geldt:

13/5=2.6

Bij een integere deling echter, krijgt u geen decimale punt, maar daarvoor in de plaats krijgt u het gedeelte van het resultaat dat geheel is, dat wil

zeggen 2 en een rest 3. Als u in MSX-BASIC 13DIV5 uitvoert, dan krijgt u:

13¥5=2 ...het getal dat heel is

Als u 13MOD5 uitrekent krijgt u:

13MOD5=3 ...de rest

De integere rekenkunde bij DIV en MOD wordt uitgevoerd met 16-bit integere waarden. Voordat de berekening wordt uitgevoerd, worden alle niet-integere waarden gewijzigd in afgekorte integere waarden, dus het afgekorte gedeelte wordt genegeerd.

16.78 ¥ 5.89 = 3 (is gelijk aan 16¥5=3)
16.78MOD5.89 = 1 (is gelijk aan 16MOD5=1)

MOD volgt op DIV (¥) in volgorde van prioriteit.
Dit voorbeeld laat de prioriteitenvolgorde zien van MOD en DIV:

```
     1580 MOD  789 DIV 10
1              789 DIV 10
                   78
2    1580 MOD      78
          20
```

Dus 1580 MOD 789¥10 = 20

Relationele bewerkers

Relationele bewerkers vergelijken twee waarden en geven een waar (-1) of niet-waar (0) als antwoord. Gewoonlijk worden zij toegepast in IF...THEN-opdrachten, maar ze kunnen ook worden opgenomen in een uitdrukking.

BEWERKER	RELATIE	VOORBEELD
=	is gelijk aan	A=B
<>	is niet gelijk aan	A<>B
<	is kleiner dan	A<B
>	is groter dan	A>B
<=	is kleiner of gelijk aan	A<=B
>=	is groter of gelijk aan	A>=B

Is de vergelijking waar dan wordt -1 afgegeven, niet-waar geeft 0 terug.
Als in een uitdrukking relationele en rekenkundige bewerkers worden gebruikt, hebben de rekenkundige bewerkers prioriteit.
Voorbeeld:

A*B=C

A*B wordt als eerste uitgevoerd, waarna het resultaat van A*B wordt vergeleken met C.
Relationele bewerkers worden uitvoerig behandeld in de sectie over IF...-THEN...ELSE (sectie 2, hoofdstuk 7).

Hoofdstuk 5

Overzicht van getallensystemen die in MSX-BASIC worden gebruikt

Omdat de meeste mensen op deze planeet tien vingers hebben, gebruiken ze het decimale getallensysteem. In tegenstelling tot de homo sapiens beschikken computers, die in principe een grote reeks schakelaars voorstellen, over twee vingers namelijk AAN en UIT (of met andere woorden, 0 of 1). Dus de computers werken intern met een binair getallensysteem. Voor het gemak van zijn gebruikers presenteert MSX-BASIC getallen in decimale notatie, maar kan natuurlijk ook binaire getallen aan in de gebruikersprogramma's. Naast binaire getallen kan MSX ook octale en hexadecimale getallen aan. Alle hexadecimale, binaire en octale getallen zijn integere waarden en moeten daarom voldoen aan de toegestane marge, dat wil zeggen tussen -32768 en 32767.
De rekenkundige functies worden uitgevoerd in het BCD-systeem (Binary Coded Decimal), dat we later nog zullen behandelen.

Binair

Definitie:	Getallensysteem met grondtal 2. Getallenvoorstelling met behulp van 0 en 1.
MSX-BASIC:	&B<binair> geeft een binaire waarde aan. BIN$ omzetting van decimaal naar binair.

Ieder cijfer van een binair getal wordt 'bit' genoemd. Acht binaire bits worden een 'byte' genoemd. Een byte is de grootte van een geheugenadres in het MSX-systeem. Een byte binaire bits kan een waarde bevatten die ligt tussen 00000000 tot 11111111, of 0 tot 255 decimaal voorgesteld.

DECIMAAL	BINAIR	IN 8-BIT
0	0	00000000
1	1	00000001
2	10	00000010
3	11	00000011
4	100	00000100

DECIMAAL	BINAIR	IN 8-BIT
5	101	00000101
6	110	00000110
7	111	00000111
8	1000	00001000
9	1001	00001001
10	1010	00001010
11	1011	00001011
12	1100	00001100
13	1101	00001101
14	1110	00001110
15	1111	00001111
16	10000	00010000

Van binair naar decimaal en van decimaal naar binair

Hoe worden 8-bit binaire waarden gewijzigd naar een decimaal equivalent?

BIT-nummer	BIJBEHORENDE WAARDE	
7	2^7	128
6	2^6	64
5	2^5	32
4	2^4	16
3	2^3	8
2	2^2	4
1	2^1	2
0	2^0	1

Voorbeeld: wijzig 10110001 in een decimale voorstelling.

```
1 0 1 1 0 0 0 1
                  1 x 2^0 =   1
                  0 x 2^1 =   0
                  0 x 2^2 =   0
                  0 x 2^3 =   0
                  1 x 2^4 =  16
                  1 x 2^5 =  32
                  0 x 2^6 =   0
                  1 x 2^7 = 128
                            ---
                            177
```

Binair Decimaal
&B10110001 = 177

Hoe wordt een decimale waarde binair voorgesteld?
Voorbeeld: stel 53 binair voor.

BIT	GETAL			
7	2^7	128		0
6	2^6	64		0
5	2^5	32	53-32=21	1
4	2^4	16	21-16=5	1
3	2^3	8		0
2	2^2	4	5-4=1	1
1	2^1	2		0
0	2^0	1	1-1=0	1

Verzamel alle binaire bits en u krijgt 53=&B00110101.
In BASIC:
Hoe wordt een 8-bit binaire waarde decimaal voorgesteld?
Voorbeeld:

```
10 X%=&B00101011
20 PRINT X%
RUN
43
```

Hoe wordt een decimale waarde binair voorgesteld?
Voorbeeld:

```
10 X%=171
20 PRINT BIN$(X%)
RUN
10101011
```

Octaal

Definitie: Een getallensysteem met als grondtal 8. Als getallenvoorstelling worden de cijfers van 0 tot en met 7 gebruikt.

MSX-BASIC: &0<octaal> stelt een octaal getal voor.
OCT$ omzetting van decimaal naar octaal.

DECIMAAL	OCTAAL
0	0
1	1
2	2
3	3
4	4

DECIMAAL	OCTAAL
5	5
6	6
7	7
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17
16	20

Omzetting van octaal naar decimaal en van decimaal naar octaal

Hoe wordt een octaal getal decimaal voorgesteld?

<decimaal>=<1ste cijfer> *8^0 + <2de cijfer> *8^1 + <3de cijfer> *8^2 + <4de cijfer> *8^3 ...

Voorbeeld: octaal &05436 naar decimaal:

6*8^0+3*8^1+4*8^2+5*8^3 = 2846

&05436 = 2846

Stel nu een decimaal getal octaal voor.
Voorbeeld: decimaal 6588 naar octaal:

Cijfer		Decimaal	Integere deling	Octaal modulus	
4	8^4	4096	6588 DIV 4096 = 1	6588 MOD 4096	= 2496
3	8^3	512	2492 DIV 512 = 4	2492 MOD 512	= 444
2	8^2	64	444 DIV 64 = 6	444 MOD 64	= 60
1	8^1	8	60 DIV 8 = 7	60 MOD 8	= 4
0	8^0	1	4 DIV 1 = 4		

Aldus verkrijgt u &014674 = 6588 (decimaal).

In BASIC

Hoe wordt een octaal getal gewijzigd in een decimaal getal?
Voorbeeld:

```
10 X%=&06517
```

```
20 PRINT X%
RUN
3407
```

Hoe wordt een decimaal getal gewijzigd in een octaal getal?
Voorbeeld:

```
10 X%=171
20 PRINT OCT$(X%)
RUN
253
```

Hexadecimaal

Definitie: Een getallensysteem met als grondtal 16. Voor de presentatie van de getallen worden de volgende cijfers/letters gebruikt: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

MSX-BASIC: &H<HEX> stelt een hexadecimaal getal voor.
HEX$ decimaal naar hexadecimaal.

DECIMAAL	HEXADECIMAAL
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10

Omzetting van hexadecimaal naar decimaal en van decimaal naar hexadecimaal

Hoe wordt een hexadecimaal getal decimaal voorgesteld?

<decimaal> = <1ste cijfer>*16^0 + <2de cijfer>*16^1 + <3de cijfer>
*16^2 + <4de cijfer>*16^3 ...

Voorbeeld: HEX &HFF naar decimaal:

(&HF)*16^0+(&HF)*16^1=(15)*1+(15)*16=255
&HFF=255

Hoe wordt een decimaal getal hexadecimaal voorgesteld?
Voorbeeld: decimaal 7752 omzetten naar een hexadecimaal getal:

Cijfer		Decimaal	Integer deling		HEX	modulus	
3	16^3	4096	7752 DIV 4096	=	1	7752 MOD 4096	= 3656
2	16^2	256	3656 DIV 256	=	14(E)	3656 MOD 256	= 72
1	16^1	16	72 DIV 16	=	4	72 MOD 16	= 8
0	16^0	1	8 DIV 1	=	8	8 MOD 1	= 0

Aldus verkrijgt u dat &H1E48 = 7752 (decimaal).

In BASIC

Hoe wordt een hexadecimaal getal gewijzigd in een decimaal getal?
Voorbeeld:

```
10 X%=&HEF3
20 PRINT X%
RUN
4189
```

Hoe wordt een decimaal getal gewijzigd in een hexadecimaal getal?
Voorbeeld:

```
10 X%=171
20 PRINT HEX$(X%)
RUN
AB
```

Opmerkingen:
1. Het binaire systeem wordt veel toegepast bij het ontwerpen van sprites, omdat het eenvoudig laat zien hoe het bit-patroon uit nullen en enen (uit en aan) bestaat.
2. Hexadecimale getallen worden gebruikt om geheugenadressen aan te geven, en instructies in machinetaal.

Overzicht van decimaal, binair, octaal en hexadecimaal:

Decimaal	Binair	Octaal	Hexadecimaal
0	0	0	0
1	1	1	1

Decimaal	Binair	Octaal	Hexadecimaal
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Het binair gecodeerd decimale systeem en de MSX

MSX beschikt over een 14-cijferige accuratesse met dubbel nauwkeurige rekenkundige BCD-functies. Dit betekent dat er geen vreemde afrondingsfouten ontstaan, zoals dit veelal het geval is bij de meeste 8-bit-computersystemen. Alle rekenkundige bewerkingen worden met deze nauwkeurigheid uitgevoerd, tenzij anders opgegeven in een gebruikersprogramma.
Waarom het gebruik van BCD? Wat zijn de voordelen en waarom is dit nauwkeuriger? Om te beginnen gaan we eens kijken hoe getallen door het binaire systeem, wat voor onze computer het meest geschikt is, worden bewerkt.
Als u in het binaire systeem een getal wilt uitdrukken dat kleiner is dan 1 (dus een getal met een decimale punt) dan krijgt u soms fouten. Er zijn bepaalde getallen die wel decimaal, maar niet binair kunnen worden uitgedrukt. Gesteld we willen 0.625 binair uitdrukken:

0.625 = 1/4 + 1/8 = 1/(2^2) = 1/(2^2)
0.625 = 2^(-2)+2^(-3)

U kunt dus blijkbaar 0.625 eenvoudig binair voorstellen.
Echter als u een getal als 0.1 binair wilt uitdrukken, dan lukt dat niet. Er is geen enkele manier om 0.1 exact uit te drukken. Het volgende laat zien waarom:

0.1 ~ 1/16 + 1/32 + 1/256 + 1/512 + 1/4096 + 1/8192 + 1/65536 ...
~ 0.09999 ...

of:

0.1 ~ 1/(2^4) + 1/2^5) +1/(2^8) + 1/(2^9) + 1/(2^13) + 1/(2^16) + ...
~ 0.0999999 ...

U kunt 0.1 niet binair voorstellen; ervoor in de plaats krijgt u een reeks. In plaats van het wijzigen van decimaal in een compleet binair systeem, gebruikt MSX het binair gecodeerd decimale systeem (BCD).
In het binaire gecodeerde systeem stelt men een decimaal getal voor met behulp van vier binaire bits. Dit betekent dat 0001 correspondeert met 1, en 0010 correspondeert met 2 enz., tot en met 1001 dat correspondeert met 9. Vier binaire bits worden niet toegepast om getallen aan te geven die liggen tussen 11 en 15, dus 1010, 1011, 1100, 1101, 1110 en 1111, worden niet gebruikt in het BCD-systeem.
Daarvoor in de plaats gebruikt 10 (decimaal) een tweede set van vier binaire bits - daarom is 10 in BCD gelijk aan 0001 0000. 1 byte (8 bits) stelt twee decimale cijfers voor.

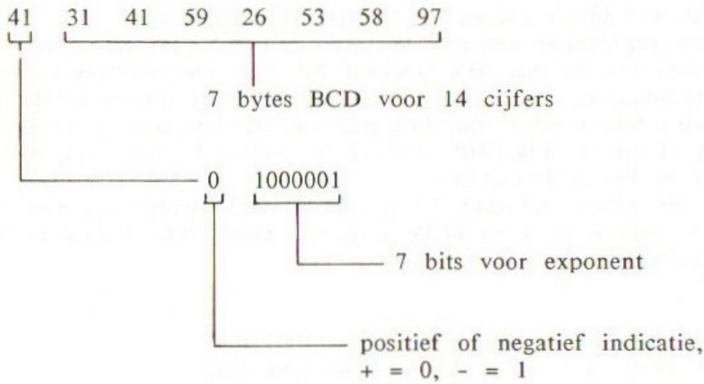
0	0000 0000	10	0001 0000 ...	90 1001 0000
1	0000 0001	11	0001 0001 ...	91 1001 0001
2	0000 0010	12	0001 0010 ...	92 1001 0010
3	0000 0011	13	0001 0011 ...	93 1001 0011
4	0000 0100	14	0001 0100 ...	94 1001 0100
5	0000 0101	15	0001 0101 ...	95 1001 0101
6	0000 0110	16	0001 0110 ...	96 1001 0110
7	0000 0111	17	0001 0111 ...	97 1001 0111
8	0000 1000	18	0001 1000 ...	98 1001 1000
9	0000 1001	19	0001 1001 ...	99 1001 1001

8 bytes dubbel nauwkeurige getallen in BCD

MSX bewaart numerieke gegevens in dubbele nauwkeurigheid, tenzij dit anders is geprogrammeerd. Dit gebeurt met een accuratesse van 14 cijfers en wordt opgeslagen in een 8 bytes binair gecodeerd decimaal systeem. De eerste byte bevat de exponent, de resterende bytes bevatten de 14 cijfers, dus 2 cijfers per byte.
Om te zien hoe de numerieke gegevens staan opgeslagen, bepalen we eerst de geheugenlocatie van die gegevens met behulp van de VARPTR-functie en tonen het resultaat in hexadecimale getallen met behulp van de PEEK-opdracht.

```
10 PI=3.1415926534890
20 ADRES=VARPTR(PI)
30 FOR I=ADRES TO ADRES+7
40 PRINT HEX$(PEEK(I)), " ";
50 NEXT I
60 PRINT
RUN
41 31 41 59 26 53 58 97
OK
```

Laten we dit eens nader bekijken:



De eerste byte geeft het positief of negatief teken en de exponent aan. Om de exponent af te leiden negeren we bit 7 en berekenen de waarde van de resterende bits, dit zijn bit 6 tot en met bit 0; we trekken er dan 65 af. Dit geeft een waarde aan die ligt tussen 1E-64 en 9.9999999999999E62.

Voordelen en nadelen van het Binair Gecodeerde Decimale Systeem (BCD)

Voordelen:

1. BCD geeft geen afrondingsfouten en kan getallen aan die binair niet mogelijk zijn, zoals bijvoorbeeld 0.1.
2. Het is bijzonder eenvoudig om een BCD-getal te tonen, omdat er geen aanpassing nodig is van binair naar decimaal. Dit verhoogt de snelheid van het afdrukproces op het beeldscherm.

Nadelen:

1. Omdat het niet een standaard binair systeem is, kost het de computer iets meer tijd, zodra het aankomt op rekenkundige bewerkingen. Echter, de Z80 Centrale Verwerkings Eenheid (CVE) beschikt over een speciale instructie DAA, afgeleid vanuit het Engels, 'Decimal Adjust Accumulator' om deze aanpassing te vereenvoudigen. DAA wijzigt de inhoud van de accumulator in packed BCD, gevolgd door optellingen of aftrekkingen met packed BCD operands.

Hoofdstuk 6

Boolean algebra (logische bewerkers)

Introductie

Hoe klein uw MSX ook mag zijn, het blijft een computer en een computer bestaat in principe uit miljoenen schakelaars. Deze kunnen AAN of UIT staan, doch de combinaties hiervan maken een werkende computer. De logica van een computer, of van deze grote hoeveelheid schakelaars, wordt geregeerd door een set van regels die vallen onder de zgn. Boolean algebra. Dit is hoofdzakelijk van toepassing op het testen van meervoudige condities die in IF...THEN-opdrachten kunnen voorkomen, met behulp van AND's en OR's, maar u kunt er ook gebruik van maken bij bitmanipulatie, hetgeen van belang is in de machinetaal.

MSX logische bewerkers

MSX biedt de correcte Boolean (logische) bewerkers, in tegenstelling tot enkele andere 8-bit microcomputers, waar deze lijken te ontbreken. Zij zijn van groot belang voor geavanceerd programmeren. Hier volgt een lijst ervan, op volgorde van prioriteit:

		Computerjargon
1	NOT	complement
2	AND	logische AND
3	OR	logische OR
4	XOR	exclusieve OR
5	EQV	equivalent
6	IMP	implicatie

NOT, AND en OR zijn de belangrijke bewerkers, omdat zij de meest gebruikte logische bewerkers zijn; en de andere bewerkers kunnen in principe worden afgeleid uit combinaties hiervan.
Logische bewerkingen worden altijd uitgevoerd op integere waarden. Ieder argument dat niet integer is, wordt gewijzigd naar een 16-bit, van teken voorzien, tweede complement integer. We zullen nu zien hoe ze werken, en

waar ze voor kunnen worden gebruikt, stap voor stap. We beginnen met NOT.

NOT

NOT is een ontkenning. Het geeft waar voor niet-waar, en niet-waar voor waar, dus, een 1 voor 0 en een 0 voor 1. Met andere woorden, het ontkent een argument. Hier volgt een waarheidstabel voor NOT.

NOT X	NOT X
0	1
1	0

NOT X 100	0000000001100100
-101	1111111110011011

AND

De Boolean algebraïsche bewerkingen van AND kunnen worden getoond in de onderstaande waarheidstabel:

AND X	Y	X AND Y
0	0	0
1	0	0
0	1	0
1	1	1

Daarom bijvoorbeeld, kan 100 AND 50 als volgt worden berekend:

X	100	0000000001100100
AND Y	50	0000000000110010
	32	0000000000100000

Voorbeeld: gebruik van AND.
De logische AND kan worden gebruikt voor het bit-testen van een bepaald getal. Bijvoorbeeld, als Y=2^n, waarin n de test-bit voorstelt die ligt tussen 0 en 7, dan X AND Y=Y als de test-bit n is 1, en X AND Y=0 als de test-bit n is 0.
BIT-TEST van de vijfde bit (n=5) voor X=117.

Y=2^5=32

X	117	0000000001110101
AND Y	32	0000000000010000
	32	0000000000010000

Daarom is de vijfde bit van het getal 117 gelijk aan 1, dus waar.

OR

Indien X, Y of beide waar zijn, geeft OR waar aan. De logische OR wordt soms logische som genoemd. De waarheidstabel voor OR is:

OR X	Y	X OR Y
0	0	0
1	0	1
0	1	1
1	1	1

Voorbeeld: 34 OR 67

X	34	0000000000100010
OR Y	67	0000000001000011
	99	0000000001100011

Voorbeeld: gebruik van logische OR.
De logische OR kunt u gebruiken voor het wijzigen van een hoofdletter in een kleine letter, omdat het verschil tussen de ASCII-codes voor hoofdletters en kleine letters een constante is, namelijk 32 (of 2^5).
Voorbeeld: verander de letter A (ASCII-code 65) in a (ASCII-code 97).

A	65	0000000001000001
OR 2^5	32	0000000000100000
a	97	0000000001100001

Deze methode is erg flexibel, omdat als u geprobeerd had om een kleine letter in een kleine letter te veranderen, er niets was gebeurd. Probeer maar.

XOR

XOR, de exclusieve OR, geeft waar (1) zodra X en Y verschillen.

XOR X	Y	X XOR Y
0	0	0
1	0	0
0	1	1
1	1	0

Voorbeeld: 100 XOR 50.

X	100	0000000001100100
XOR Y	50	0000000000110010
	86	0000000001010110

XOR kan ook worden bereikt met behulp van NOT, AND en OR. Hier volgt de manier waarop dit geschiedt:

X XOR Y = (X AND (NOT Y)) OR ((NOT X) AND Y

Dus, in duidelijke taal, XOR is zowel X en niet Y als niet X en Y.

X XOR Y = (X AND (NOT Y)) OR ((NOT X) AND Y)

Bewijs:

X	Y	NOT X	NOT Y
0	0	1	1
1	0	0	1
0	1	1	0
1	1	0	0

X	(NOT Y)	X AND (NOT Y)
0	1	0
1	1	1
0	0	0
1	0	0

(NOT X)	Y	(NOT X) AND Y
1	0	0
0	0	0
1	1	1
0	1	0

(X AND (NOT Y))	((NOT X) AND Y	(X AND (NOT Y)) OR ((NOT X) AND Y)
0	0	0
1	0	1
0	1	1
0	0	0

Daarom geldt: X XOR Y = (X AND (NOT Y)) OR ((NOT X)) AND Y)
Voorbeeld: het gebruik van XOR.
XOR beschikt over de unieke eigenschap, dat als u een waarde tot tweemaal toe bewerkt met een XOR van een andere waarde, de oorspronkelijke waarde weer is hersteld.

X XOR Y XOR Y = X

Laten we eens als voorbeeld proberen 100 XOR 50 XOR 50.

100 XOR 50

	X 100	0000000001100100
XOR	Y 50	0000000000110010
	86	0000000001010110

(100 XOR 50) XOR 50

	X 86	0000000001010110
XOR	Y 50	0000000000110010
	100	0000000000110010

Daarom is 100 XOR 50 XOR 50 = 100.

EQV

Dit is de logische equivalentie-functie. EQV geeft waar voor iedere X die gelijk is aan Y, en niet-waar indien ze verschillen.

EQV X	Y	X EQV Y
0	0	1
1	0	0
0	1	0
1	1	1

Voorbeeld: 51 EQV 75.

	X 51	0000000000110011
EQV	Y 74	0000000001001010
	-122	1111111110000110

De volgende relatie is waar:

X EQV Y = (X AND Y) OR ((NOT X) AND (NOT Y))

Bewijs:

X	Y	NOT X	NOT Y
0	0	1	1
1	0	0	1
0	1	1	0
1	1	0	0

X	Y	X AND Y
0	0	0
1	0	0
0	1	0
1	1	1

(NOT X)	(NOT Y)	((NOT X) AND (NOT Y))
1	1	1
0	1	0
1	0	0
0	0	0

(X AND Y)	((NOT X) AND (NOT Y))	(X AND Y) OR ((NOT X) AND (NOT Y))
0	1	1
0	0	0
0	0	0
1	0	1

Daarom is X EQV Y = (X AND Y) OR ((NOT X) AND (NOT Y))

IMP

IMP-waarheidstabel:

IMP	X	Y	X IMP Y
	0	0	1
	1	0	0
	0	1	1
	1	1	1

100 IMP 50 kan als volgt worden berekend:

```
     X 100     0000000001100100
IMP  Y  50     0000000000110010
       -69     1111111110111011
```

De volgende relatie is waar:

X IMP Y = (NOT X) OR Y

Bewijs:

X	Y	(NOT X) OR Y
1	0	1
0	0	0
1	1	1
0	1	1

Daarom geldt X IMP Y = (NOT X) OR Y

Wel, dat was alles voor zover het MSX-BASIC betreft, maar BOOLEAN algebra houdt hier niet op. Er zijn verscheidene andere functies, zoals NOR en NAND, die niet zijn opgenomen in MSX-BASIC, doch die kunnen worden gesimu-

leerd met behulp van NOT, AND en OR. Onthoud dat deze drie de basis vormen van BOOLEAN algebra, en dat ze altijd kunnen worden gecombineerd om de andere functies te bereiken.
Bekijk de tabel hieronder eens. Twee binaire waarden, X en Y, kunnen ieder een waarde 1 of 0 aannemen. Ze kunnen vier verschillende combinaties aannemen, omdat 2^2=4. Er zijn 16 combinaties van deze combinaties (of functies) omdat 2^2^2=16. Aldus verkrijgt u een lijst van alle mogelijke combinaties of waarheidstabellen.

X	Y	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
0	0	0	0	1	0	1	0	0	0	1	0	1	1	1	0	1	1
0	1	0	0	1	1	0	0	0	1	0	1	1	0	1	1	0	1
1	0	0	1	0	0	1	0	1	0	0	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1	0	0	0	1	1	1	0	0	1	1

U kunt het volgende herkennen:

f2	is	NOT
f5	is	AND
f9	is	OR
f13	is	XOR
f10	is	IMP
f14	is	EQV

Deze zijn allemaal aanwezig in MSX-BASIC, maar wat doen we met de andere, dus f1 en f3 enz.?
In deze sectie zullen we de niet-MSX logische bewerkers nader bekijken. Deze kunnen niet zoals ze zijn, worden toegepast in MSX, daarom wordt een equivalent met behulp van NOT, AND en OR gegeven.
f0 geeft altijd niet-waar, welke combinatie van X en Y ook wordt gebruikt. Deze wordt niet gebruikt in computersystemen.

f1	Is altijd X, welke waarde Y ook aanneemt.
f2	NOT X (zie ook NOT).
f3	Is altijd Y, welke waarde X ook aanneemt.
f4	NOT Y (zie ook NOT).
f5	X AND Y (zie ook AND).
f6	NOR-functie. De MSX-equivalenten zijn: (NOT X) AND (NOT Y) of als alternatief NOT (X OR Y) ... zie de wet van De Morgan.
f9	X OR Y (zie logische OR).
f10	((NOT X) OR Y) Identiek aan IMP.
f11	X IMP Y (zie IMP). Identiek aan f10. Als alternatief X OR (NOT Y).
f12	NAND-functie. Het is de ontkenning van AND (NAND staat voor NOT AND). NOT (X AND Y) (NOT X) OR (NOT Y)

	X NAND Y = (NOT X) OR (NOT Y) = NOT (X AND Y) Zie de wet van De Morgan, verderop.
f13	X OR Y. Exclusieve OR (zie XOR). Als alternatief (X AND (NOT Y)) OR ((NOT X) AND Y).
f14	X EQV Y. Logische equivalentie. X EQV Y = (X AND Y) OR ((NOT X) AND (NOT Y)).
f15	altijd waar voor iedere waarde van X en Y. Wordt niet gebruikt.

NAND en NOR, beide niet aanwezig in MSX-BASIC, zijn handige functies om te onthouden.

Enkele nuttige logische relaties

Er is een variëteit aan logische relaties welke kunnen worden gebruikt in Boolean algebra. Zij kunnen worden gebruikt in het verkorten van breedvoerige logische uitdrukkingen door middel van minimalisering.
Gebruik van logische OR:

1 X OR 0=X
2 X OR X=X
3 X OR X OR X OR X OR ... =X
4 X OR (NOT X)=1
5 X OR 1=1
6 X OR (X AND Y)=X
7 X OR ((NOT X) AND Y)=X OR Y

Gebruik van logische AND:

1 X AND 0=0
2 X AND X=X
3 X AND X AND X AND X ... =X
4 X AND (NOT X)=0
5 X AND 1=X
6 X AND (X AND Y)=X AND Y
7 X AND ((NOT X) AND Y)=0

Voorbeelden:
Probeer enkele van de volgende voorbeelden uit te voeren op uw computer en controleer of de bovenstaande relaties inderdaad waar zijn.

```
                                        ANTWOORD

PRINT 145 OR 0                          145
PRINT 167 OR 167                        167
PRINT 167 OR 167 OR 167                 167
PRINT 133 OR (133 AND 222)              133
PRINT 111 AND 0                         0
```

```
PRINT 111 AND 111                          111
PRINT 111 AND 111 AND 111 AND 111          111
PRINT 234 AND (NOT 234)                    0
PRINT 234 AND (234 OR 54)                  234
```

De regels van De Morgan

Naast de bovengenoemde relaties, beschikt Boolean algebra over twee speciale trucs om de lengte van logische uitdrukkingen te reduceren.

1. Ontkenning van logische OR.
   De ontkenning (NOT) van een logische OR is equivalent aan de logische AND van de ontkenningen van de variabelen waaruit de logische OR is opgebouwd. Dit kan worden uitgedrukt als:

   NOT (X OR Y) = (NOT X) AND (NOT Y)

2. Ontkenning van logische AND.
   De ontkenning van een logische AND is equivalent aan de logische OR van de ontkenningen van de variabelen waaruit de logische AND is opgebouwd. Dit kan worden voorgesteld als:

   NOT (X AND Y) = (NOT X) OR (NOT Y)

Wetten van het opnieuw schikken

In de wiskunde bestaat een aantal wetten, die u toestaan om complexe uitdrukkingen opnieuw te schikken en zodoende te vereenvoudigen. Zij staan bekend als de commutatieve, associatieve en distributieve wetten. Boolean algebra heeft gelijksoortige wetten, en wel als volgt:

1. Commutatieve wetten
   a. X OR Y = Y OR X
   b. X AND Y = Y AND X

2. Associatieve wetten
   a. X AND (Y AND Z) = (X AND Y) AND Z = X AND Y AND Z
   b. X OR (Y OR Z) = (X OR Y) OR Z = X OR Y OR Z

3. Distributieve wetten
   a. X AND (Y OR Z) = (X AND Y) OR (X AND Z)
   b. X OR (Y AND Z) = (X OR Y) AND (X OR Z)

Overzicht van MSX logische bewerkers waarheidstabellen

X	Y	AND	OR	XOR	EQV	IMP
0	0	0	0	0	1	1
1	0	0	1	1	0	0
0	1	0	1	1	0	1
1	1	1	1	0	1	1

Hoofdstuk 7

Boolean II: De IF ... THEN ... ELSE

Wanneer u een programma aan het schrijven bent, is het zeer waarschijnlijk dat de computer beslissingen moet nemen, terwijl uw programma wordt uitgevoerd. Om op bepaalde condities beslissingen te nemen, kunt u gebruik maken van de IF-opdracht om de condities te testen, waarna afhankelijk van het resultaat een bepaalde taak wordt uitgevoerd. IF wordt altijd samen met THEN uitgevoerd en soms ook met het ELSE-sleutelwoord. In principe zijn er twee soorten formaten:

IF <conditie> THEN <opdracht>

en

IF <conditie> THEN <opdracht> ELSE <opdracht>

IF wordt gevolgd door de te testen condities. Als de condities waar blijken te zijn, worden de opdrachten die volgen op THEN uitgevoerd. Zijn de condities niet waar, dan worden, als er geen ELSE is opgegeven in de regel, de opdracht(en) van de volgende regel uitgevoerd, zodat alle opdrachten direct achter THEN worden overgeslagen.
Blijkt er echter wel een ELSE-opdracht aanwezig te zijn in de regel en zijn de condities niet waar gebleken, dan worden de opdrachten achter ELSE uitgevoerd. In dit geval worden de opdrachten die staan tussen THEN en ELSE volledig genegeerd.
Om te beginnen zullen we de condities die deel zijn van de IF...THEN... ELSE-structuur, nader bekijken.

<CONDITIES>

Dit gedeelte volgt alle regels en afspraken zoals beschreven in de hoofdstukken 'Uitdrukkingen en bewerkers' en 'Boolean algebra'. Deze sectie toont de praktische kant van wat werd gezegd in deze laatste twee hoofdstukken.

Numerieke vergelijking

De eenvoudigste conditie vergelijkt twee waarden, dus er kan worden gecontroleerd of de ene waarde groter is dan de andere. Er wordt ook gebruik gemaakt van de relationele bewerkers, zoals <, >, <>, =, <= en >=.
Voorbeelden:

1.	15<8	geeft	0 (niet waar)
2.	a%=67	geeft	-1 (waar) als a%=67
			0 (niet waar) als a%<>67

In een IF...THEN-formaat:

```
1.  IF D%=100 THEN PRINT "D% IS 100"
2.  IF V<=G THEN PRINT "V<=G"
```

Dit zijn de eenvoudigste vormen, maar u mag ook rekenkundige uitdrukkingen gebruiken aan beide zijden van de relationele bewerker.

```
1.  IF F%*H%=9876 THEN PRINT "WAAR"
2.  IF 2^8=N%-19*Y% THEN PRINT "WAAR"
```

Zo is er ook geen reden om geen gebruik te maken van BASIC-functies als COS en INSTR enz. In feite kunt u ook een door uzelf gedefinieerde functie opnemen.

```
1.  IF INSTR(A$,B$)=5 THEN PRINT B$
2.  IF COS(0.6242)>=TAN(D%) THEN PRINT "WAAR"
3.  IF FNA(D%*I%-100)=&HFF THEN PRINT "FF"
```

Numerieke conditie zonder vergelijking

Het is niet strikt noodzakelijk om een relatie te gebruiken als conditie. Het kan in feite ook een enkelvoudige numerieke variabele zijn of een eenvoudige uitdrukking. Kijk eens naar het volgende voorbeeld:

```
IF X% THEN PRINT "X%=WAAR"
```

In dit geval veronderstelt de computer de conditie als waar als de variabele X% ongelijk aan nul is, en als niet-waar als X%=0.
In het algemeen:

IF <uitdrukking> THEN <opdracht>

<uitdrukking>=WAAR als <uitdrukking> niet nul teruggeeft
<uitdrukking>=NIET WAAR als <uitdrukking> nul teruggeeft

U kunt hier in een aantal situaties gebruik van maken. Hier volgen enkele

voorbeelden om dit punt te illustreren.

1. `IF A<>0 THEN PRINT "A NOT 0"`
   ... kan zijn
   `IF A THEN PRINT "A NOT 0"`

2. Om te controleren op de aanwezigheid van een bepaald teken in een string:

   ```
   IF INSTR(S$,"A") THEN PRINT "S$ BEVAT LETTER A"
   ```

STRING-vergelijking

U mag twee strings vergelijken met behulp van relationele bewerkers op vrijwel dezelfde manier. De vergelijking geschiedt door telkens een teken van iedere string te nemen en daarvan de ASCII-codes te vergelijken. Als de ASCII-codes verschillen, gaat de kleine lettervorm voor de hoofdlettervorm. Om te bepalen welke ASCII-code behoort bij welk teken, kunt u de ASCII-tabel raadplegen. Als tijdens het vergelijken van strings het einde van een string is bereikt, wordt de kortste string als lager verondersteld. String-vergelijking kan worden gebruikt om strings op alfabetische volgorde te zetten.
In het volgende voorbeeld zijn alle condities waar.

1. `"abc"="abc"`
2. `"ABC"<"abc"`
3. `"ABC"="ABD"`
4. `"ab"<"abc"`
5. `"1234"<"12345"`

Het testen van meervoudige condities met Boolean bewerkers

Het is mogelijk om meervoudige condities te testen in een opdracht, door gebruik te maken van de logische Boolean bewerkers. Er bestaan zes logische bewerkers in MSX-BASIC, maar slechts drie ervan, NOT, AND en OR zijn relevant. Zij voldoen aan alle regels die de Boolean algebra regeren, zoals de wetten van De Morgan.

1. Eenvoudig gebruik van AND:

   IF <conditie-1> AND <conditie-2> THEN <opdrachten>

   In dit geval zal de computer alleen de opdrachten achter THEN uitvoeren als de condities waar zijn, dus:

   ```
   IF X=0 AND Y$="JA" THEN PRINT "MOOI ZO"
   ```

2. Eenvoudig gebruik van OR:

IF <conditie-1> OR <conditie-2> THEN <opdrachten>

In dit geval zal de computer de opdrachten uitvoeren als één of beide condities waar zijn, dus:

```
IF X=0 OR Y=0 THEN PRINT "EEN VAN BEIDE IS NUL"
```

3. Eenvoudig gebruik van NOT:

IF NOT <conditie> THEN <opdrachten>

Indien de conditie waar is, resulteert NOT <conditie> in het tegenovergestelde, dus niet-waar, en vice versa, dus:

```
IF NOT (V=U*8) THEN PRINT "V NOT U*8"
```

Het zal u reeds zijn opgevallen dat het zinloos is om NOT toe te passen in de bovengenoemde relatie, als u het kunt herschrijven tot:

```
IF V<>U*8 THEN PRINT "V NOT U*8"
```

Hier volgt een lijst van dergelijke relaties die dezelfde betekenis hebben:

NOT (X=Y)	X<>Y
NOT (X<>Y)	X=Y
NOT (X<=Y)	X>Y
NOT (X>=Y)	X<Y
NOT (X>Y)	X<=Y
NOT (Y<Y)	X>=Y

Dus, verspil geen geheugenruimte.
Wat NOT kan bereiken is het tegenovergestelde resultaat van een uitdrukking als:

```
IF NOT INSTR(A$,"j") THEN PRINT "j is NOT INSTRing A$"
```

Bovenstaand voorbeeld is zelfverklarend.

Minimalisering - de wet van De Morgan

Ontkenning van de logische OR.
De volgende drie opdrachten hebben een gelijk effect:

```
IF NOT (X=9 OR Y=10)
IF NOT (X=9) AND NOT (Y=10)
```

```
IF NOT (X=9) AND NOT (Y=10)
IF X<>9 AND Y<>10
```

Ontkenning van de logische AND.
De volgende drie opdrachten hebben een gelijk effect:

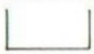
```
IF NOT (X=9 AND Y=10)
IF NOT (X=9) OR NOT (Y=10)
IF X<>9 OR Y<>10
```

In geval van twijfel moet u ze uitvoeren en zelf het resultaat bekijken.

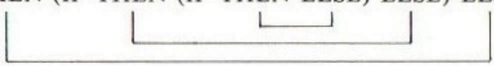
IF...THEN...ELSE-structuren

ELSE is een deel van de IF...THEN...ELSE-structuur. ELSE geeft de computer aan dat, indien de conditie van IF niet voldoet (dus niet-waar is), de opdrachten na THEN dienen te worden overgeslagen en de opdrachten die volgen op ELSE moeten worden uitgevoerd.
Meervoudige IF...THEN...ELSE-opdrachten kunnen worden gemaakt, evenals het nesten van IF...THEN...ELSE-structuren. De beperking bestaat alleen uit de maximumlengte van een regel, welke 255 tekens bevat. Indien er minder ELSE-sleutelwoorden zijn dan THEN's, wordt iedere ELSE vergeleken met de dichtstbij gelegen THEN.

IF THEN (IF THEN ELSE)



IF THEN (IF THEN (IF THEN ELSE) ELSE) ELSE



Meervoudige IF...THEN...ELSE:

IF THEN ELSE IF THEN ELSE

De tweede IF wordt uitgevoerd, indien de eerste niet-waar blijkt te zijn.
Syntax:
IF...THEN GOTO <regel>-vorm kan op twee manieren worden afgekort:

1. IF...THEN <regel>
2. IF...GOTO <regel>

Ook ELSE GOTO <regel> kan worden afgekort:

1. IF...THEN...ELSE <regel>

2. IF...THEN...GOTO <regel>

Punten om te onthouden

IF...THEN...ELSE-opdrachten neigen al gauw te lang te worden voor een regel, omdat u meerdere opdrachten kunt meegeven achter THEN en ELSE. Als dit het geval is, is het aan te raden gebruik te maken van subroutines, met behulp van GOSUB.
Het is aan te bevelen geneste IF...THEN...ELSE-opdrachten te vermijden, omdat deze tot een warboel kunnen leiden. Als u niet oplet, creëert u al gauw een spaghetti-programma.

Hoofdstuk 8

PRINT USING-opdracht

Inleiding

MSX beschikt over een uitgebreide besturing van het afdrukformaat, met behulp van PRINT USING. De PRINT USING-opdracht stelt u in staat om strings of getallen op verschillende manieren af te drukken.

Syntax

PRINT USING <string-exp> ; <lijst van termen>

Er bevinden zich twee gedeelten in het argument van PRINT USING, de string die het formaat specificeert en de lijst van termen die moeten worden afgedrukt. <string-exp> bevat de string die het speciale formaat aangeeft en bepaalt hoe de lijst met termen moet worden afgedrukt. Het kan zowel een string als een string-variabele zijn.

Verklaring van de tekens

! Het uitroepteken geeft aan dat alleen de eerste positie dient te worden afgedrukt.
Hier volgt een voorbeeld, waarbij alleen de initiaal van elk van de voornamen en achternamen die in de DATA-regels staan wordt afgedrukt.

```
10 FOR I=1 TO 3
20 READ VOORNM$,ACHTERNM$
30 PRINT USING "!";VOORNM$,".",ACHTERNM$
40 NEXT I
50 DATA ROBERT,KLAASSEN
60 DATA KLAAS,VAAK
70 DATA ANNIE,VERHAGEN
RUN
R.K
K.V
A.V
```

@ Voegt een opgegeven string toe op de positie aangegeven door @.

```
10 A$="QWERTY"
20 PRINT USING "DIT IS EEN @ TOETSENBORD",A$
RUN
DIT IS EEN QWERTY TOETSENBORD
```

\# De # geeft aan dat een cijfer moet worden afgedrukt. Bijvoorbeeld: PRINT USING "#.###";1.3 drukt af:

1.300

dus # geeft het formaat aan van het af te drukken getal.
Zoals u kunt zien in bovenstaand voorbeeld, mag u een decimale punt meegeven in het formaat van de string, tussen de #-symbolen. Blijkt het uiteindelijke getal minder cijfers te bevatten dan is opgegeven door de formaattekens, dan wordt naar rechts aangepast met nullen en vooraan aangevuld met spaties.

```
10 PRINT "####.#######"
20 PRINT USING "####.#######";ATN(1)*4
RUN
####.#######
   3.1459265
```

Als de formaattekens meer decimale posities aangeven dan nodig blijkt, wordt de ruimte opgevuld door nullen.

```
10 PRINT "##.######"
20 PRINT USING "##.######";99.999
RUN
##.######
99.999000
```

Getallen worden afgerond als ze niet passen in het opgegeven veld.

```
10 PRINT "##.###"
20 PRINT USING "##.###";10.2367
RUN
##.###
10.237
```

U mag twee of meer getallen afdrukken met hetzelfde formaat in een PRINT USING-opdracht.

```
PRINT USING "##.##",9.866,18.7
9.97     18.70
```

Als het af te drukken getal negatief is, wordt het negatiefteken

getoond, maar toch een positie in de specificatie gebruikt.

```
10 PRINT "###.####"
20 PRINT USING "###.####";-2.63
RUN
###.####
 -2.6300
```

Als het af te drukken getal niet in het veld met de formaattekens past, wordt voor het getal een %-teken afgedrukt. Dit gebeurt ook als het afgeronde getal buiten de grootte van het veld valt.

```
PRINT USING "#.###",9.9999
%10.000
```

\+ Een plusteken ('+'), dat staat aan het begin of aan het einde van de formaatstring, resulteert in de afdruk van het teken van het getal, dus bijvoorbeeld een '+' of '-' op de opgegeven plaats.

```
PRINT USING "+###.#####";-0.123422,10.986
-0.12342   +10.98600
PRINT USING "####.#+";10.71,100,-200.5
10.7+         100.0+   200.5-
```

** De dubbele asterisk geeft aan dat spaties aan het begin van een numeriek veld moeten worden opgevuld met sterretjes ('*'). Ook specificeren zij twee posities in het veld.

```
10 PRINT "**##.########"
20 PRINT USING "**##.########";ATN(1)*4
30 PRINT USING "**##.########";-ATN(1)*4
RUN
**##.########
***3.14159265
**-3.14159265
```

$$ De dubbele dollartekens veroorzaken dat er een dollarteken wordt geplaatst aan het begin van een getal. Ook $$ specificeert twee posities in het formaatveld, waarvan één de '$' is.

```
10 PRINT "$$#.########"
20 PRINT USING "$$#.########";ATN(1)*4
30 PRINT USING "**#.########";-ATN(1)*4
RUN
$$#.########
 $3.14159265
-$3.14159265
```

Let er op dat u geen exponentieel formaat kunt gebruiken samen met de

dollartekens.

**$ Dit teken is een combinatie van de **- en $$-tekens.
Alle spaties worden opgevuld met * en het getal wordt voorafgegaan door een $-teken. **$ geeft ook drie posities aan in het formaatveld, waarvan de '$' er één is.

```
10 PRINT "**$#.########"
20 PRINT USING "**$#.########";ATN(1)*4
30 PRINT USING "**$#.########";-ATN(1)*4
RUN
**$#.########
*-$3.14159265
```

, Een komma wordt geplaatst links voor de decimale punt in de formaatstring en zorgt er voor dat per drie cijfers een komma wordt afgedrukt.

```
10 PRINT "##########,.##"
20 PRINT USING "##########,.##";1090382.88#
30 PRINT "$$##########,.##"
40 PRINT USING "$$##########,.##";1090382.88#
RUN
##########,.##
  1,090.382.88
$$##########,.##
   $1,090,382.88
```

Een komma aan het einde van de formaatstring drukt een komma aan het einde van het getal af.

```
PRINT USING "##.##,";12.567
12.57,
```

^^^^ Dit is de specificatie voor een exponent en biedt ruimte voor E+xx.
U kunt ook de positie van de decimale punt bepalen. Significante cijfers worden naar links aangepast en de exponent wordt eveneens aangepast.

```
PRINT USING "##.##^^^^";200.00
 2.00e+02
PRINT USING "+#.##^^^^";200.00
+2.00e+02
PRINT USING "#.##^^^^+";-200.00
2.00E+02-
```

Aantekeningen

Als het aantal gespecificeerde cijfers meer dan 24 bedraagt, wordt een foutmelding gegeven.

LPRINT USING

LPRINT USING is gelijk aan PRINT USING maar drukt alles af op de printer en niet op het scherm.

PRINT # USING

PRINT # USING is exact gelijk aan PRINT USING, maar nu wordt naar de periferie geschreven die is aangegeven door het bestandsnummer na het #-teken. U moet eerst een OPEN-opdracht uitvoeren om een channel (kanaal) te openen naar de door u gekozen periferie, alvorens u deze opdracht kunt laten uitvoeren.

Syntax

PRINT #<bestandsnummer>,USING<string-exp>;<lijst met uitdrukkingen>

Hoofdstuk 9

Evenementbehandeling en interruptie door BASIC

Inleiding

De MSX-computer is uitgerust met verscheidene evenement-behandelings-opdrachten. Evenement-behandeling wordt bestuurd door middel van een interruptie. Dit wil zeggen dat, tijdens de uitvoering van een programma, de computer ook let op het gebeuren van een evenement. Indien dat evenement plaatsvindt, wordt de programma-uitvoering onderbroken door direct te springen naar een vooraf bepaalde gebruikerssubroutine.
De MSX interruppeert voor de volgende evenementen:

1.	zet tijdinterval	ON INTERVAL GOSUB
2.	indrukken van functietoets	ON KEY GOSUB
3.	<CTRL> <STOP>	ON STOP GOSUB
4.	indrukken van trekker	ON STRIG GOSUB
5.	botsing van een SPRITE	ON SPRITE GOSUB
6.	fouten	ON ERROR GOSUB

1. Tijdinterval-interruptie

Gebruikte opdrachten:

ON INTERVAL = <tijd> GOSUB <regel>
INTERVAL ON/OFF/STOP

De MSX beschikt over zijn eigen interne klok, die wordt gestart zodra u de computer aanschakelt. De tijd kan worden gevonden met behulp van de TIME-functie. TIME wordt iedere 1/50ste van een seconde verhoogd, de Video Display Processor veroorzaakt dan telkens een interruptie.
Met behulp van de ON INTERVAL GOSUB-opdracht kunt u een tijdsinterval opgeven voor een op te treden interruptie. Omdat de TIME iedere 1/50ste seconde wordt verhoogd, moet u voor het bereiken van een interruptie na 1 seconde INTERVAL gelijkstellen aan 50, dus ON INTERVAL=50 GOSUB <regel>.
ON INTERVAL = <tijd> GOSUB geeft ook aan welke subroutine dient te worden aangeroepen ingeval de interruptie optreedt. De computer springt naar deze

subroutine zodra de interrupt optreedt, zonder zich iets aan te trekken van waar hij op dat moment mee bezig was.
De interval-interruptie wordt geactiveerd met INTERVAL ON. Dit vertelt de computer dat begonnen kan worden met het tellen van de klok. Tenzij deze opdracht wordt uitgevoerd, werkt de ON INTERVAL GOSUB niet.
Zodra de tijdinterruptie optreedt, wordt er een automatische INTERVAL STOP uitgevoerd. Dit voorkomt dat er een tijdinterruptie optreedt gedurende de huidige tijdinterruptie-subroutine. Echter, er wordt wel onthouden dat er een interruptie is opgetreden gedurende de huidige subroutine en de computer zal daarom de subroutine meteen opnieuw uitvoeren zodra dit het geval is, tenzij de huidige subroutine de tijdinterruptie de-activeert met behulp van INTERVAL OFF.
Na het verlaten van de interruptie-subroutine, voert de computer automatisch een INTERVAL ON uit, om zodoende interrupties weer mogelijk te maken, tenzij een INTERVAL OFF wordt uitgevoerd binnen in de subroutine.
U kunt op ieder willekeurig moment de tijdinterrupt de-activeren met behulp van INTERVAL OFF.
De tijdinterrupt wordt gede-activeerd zodra een BASIC-programma wordt verlaten, dus een interruptie tijdens de directe mode is onmogelijk. Dit gebeurt ook als er een fout wordt geconstateerd.
Voorbeelden:
Een kort programma dat een digitale klok met bliep-signaal simuleert:

```
10 ON INTERVAL=50 GOSUB 60
20 INTERVAL ON
30 CLS
40 S=0:M=0
50 GOTO 50
55 REM INTERVAL SUBROUTINE
60 IF S=60 THEN S=0:M=M+1:LOCATE 10,11:PRINT "MIN";M:BE
EP
70 S=S+1
80 LOCATE 10,10:PRINT "SEC";S
90 BEEP
100 RETURN
```

REGEL 10 Zet tijdinterval op 1 seconde met subroutine op 60
REGEL 20 Activeert detectie van interval
REGEL 30 Maakt het beeldscherm schoon
REGEL 40 Initialiseert S en M
REGEL 50 Oneindige lus die op interruptie wacht
REGEL 60 60 seconden is één minuut
REGEL 70 S+1 seconde
REGEL 80 Drukt de seconden af
REGEL 90 De bliep
REGEL 100 Keert terug naar de hoofdroutine, in dit geval altijd regel 50

SEC 54	geeft iedere seconde een enkele bliep en
MIN 3	iedere minuut een dubbele bliep

2. Interruptie door een functietoets

Gebruikte opdrachten:

ON KEY GOSUB <regel> , <regel>, ...
KEY(<keynummer>) ON/OFF/STOP

Het is mogelijk om interrupts op te zetten voor alle tien de functietoetsen met behulp van de ON KEY GOSUB- en KEY( ) ON/OFF/STOP-opdrachten. ON KEY GOSUB wordt gevolgd door een lijst van regelnummers, die ieder de subroutine aangeven voor iedere functietoets. Als er geen corresponderende subroutine bestaat voor een functietoets, kan het regelnummer achterwege blijven.
KEY(<toetsnummer>) ON-opdrachten activeren functietoets-interrupties voor iedere functietoets. Tenzij deze opdracht is uitgevoerd, zal de computer geen interruptie uitvoeren voor een functietoets. Is er sprake van een interruptie, dan springt de computer naar de opgegeven subroutine, met behulp van ON KEY GOSUB.
Zodra de functietoets-interrupties optreden, wordt er automatisch een KEY(<toetsnummer>) STOP uitgevoerd. Dit voorkomt dat er een interruptie optreedt voor dezelfde functietoets tijdens de huidige interruptie-subroutine. Echter, er wordt wel onthouden dat de functietoets is ingedrukt gedurende de uitvoering van de subroutine. De computer zal daarom in dit geval direct dezelfde subroutine weer uitvoeren, zodra de huidige subroutine is afgerond, tenzij de huidige subroutine de KEY-interruptie deactiveert door het uitvoeren van een KEY(<toetsnummer>) OFF.
Na het verlaten van de subroutine voert de computer automatisch een KEY ON uit, om interrupties weer mogelijk te maken, tenzij KEY(<toetsnummer>) OFF werd uitgevoerd door de subroutine.
De KEY-interruptie wordt gede-activeerd als het programma niet draait en ook als er een fout wordt geconstateerd.
Voorbeeld:

```
10 ON KEY GOSUB 100,120,140,160,180,200
20 KEY(1) ON
30 KEY(2) ON
40 KEY(3) ON
50 KEY(4) ON
60 KEY(5) ON
70 KEY(6) ON
80 GOTO 80
90 REM SUBROUTINES
```

```
100 PRINT "FUNKTIE 1"
110 RETURN
120 PRINT "FUNKTIE 2"
130 RETURN
140 PRINT "FUNKTIE 3"
150 RETURN
160 PRINT "FUNKTIE 4"
170 RETURN
180 PRINT "FUNKTIE 5"
190 RETURN
200 PRINT "FUNKTIE 6"
210 RETURN
```

REGEL 10 Zet subroutine voor F1, F2, F3, F4, F5 en F6
REGEL 20 Activeert functietoets F1
REGEL 30 Activeert functietoets F2
REGEL 40 Activeert functietoets F3
REGEL 50 Activeert functietoets F4
REGEL 60 Activeert functietoets F5
REGEL 70 Activeert functietoets F6
REGEL 80 Oneindige lus die wacht op een functietoets
REGEL 100 Routine voor F1
REGEL 110 Keert terug naar waar vandaan werd aangeroepen
REGEL 120 Routine voor F2
REGEL 130 Keert terug naar waar vandaan werd aangeroepen
REGEL 140 Routine voor F3
REGEL 150 Keert terug naar waar vandaan werd aangeroepen
REGEL 160 Routine voor F4
REGEL 170 Keert terug naar waar vandaan werd aangeroepen
REGEL 180 Routine voor F5
REGEL 190 Keert terug naar waar vandaan werd aangeroepen
REGEL 200 Routine voor F6
REGEL 210 Keert terug naar waar vandaan werd aangeroepen

3. <CTRL> <STOP>-interruptie en kraak-veilig maken

Gebruikte opdrachten:

ON STOP GOSUB <regel>
STOP ON/OFF/STOP

Een BASIC-programma kan erg eenvoudig worden gekraakt. Het enige dat u hoeft te doen, is indrukken van <CTRL> <STOP>. Dit onderbreekt het programma direct en plaatst u in de commando-mode. Echter, het kan voorkomen dat u uw programma 'kraak-veilig' wilt maken, om te voorkomen dat andere gebruikers de inhoud van uw programma kunnen zien. Dit kan worden bereikt door het detecteren van <CTRL> <STOP> met ON STOP GOSUB.
De ON STOP GOSUB-opdracht zet de subroutine voor de <CTRL> <STOP>-inter-

ruptie. Met deze subroutine kunt u of een boodschap 'Onderbreek dit programma niet', of alleen een RETURN-opdracht plaatsen, zodat het programma onmiddellijk doorgaat ook al werd <CTRL> <STOP> ingedrukt.
STOP ON/OFF/STOP activeert/de-activeert <CTRL> <STOP>-detectie.
Als STOP ON wordt uitgevoerd, begint BASIC te controleren of <CTRL> <STOP> wordt ingetikt, na iedere uitgevoerde opdracht. Wordt een <CTRL> <STOP> gedetecteerd, dan wordt BASIC gedirigeerd naar de subroutine die stond aangegeven in de ON GOSUB STOP, die eerder werd uitgevoerd.
Zodra de interruptie optreedt wordt een automatische STOP STOP uitgevoerd. Dit voorkomt het optreden van een interruptie gedurende de huidige interruptie-subroutine. Echter, er wordt wel onthouden dat <CTRL> <STOP> werd ingedrukt gedurende de huidige subroutine, de computer zal in dat geval direct beginnen met het opnieuw uitvoeren van de subroutine, zodra de huidige subroutine is beëindigd, tenzij de huidige subroutine de <CTRL> <STOP>-interruptie de-activeert door het uitvoeren van een STOP OFF. Na het verlaten van de interruptie-routine zal de computer automatisch een STOP ON uitvoeren, tenzij een STOP OFF werd uitgevoerd door de subroutine.
De enige manier om een 'kraak-veilig' programma toch te kraken, is door het systeem te 'resetten' met behulp van de zogenaamde 'RESET'-knop op uw MSX-computer. Daarom moet u onthouden dat u uw 'kraak-veilige' programma eerst wegschrijft naar tape of floppy, voordat u een RUN geeft.
Voorbeelden:
Hier volgt een kant en klare 'kraak-veilige' routine die u aan uw programma's kunt toevoegen.

```
10 ON STOP GOSUB 10000
20 STOP ON
...
...
    Voeg hier uw programma toe
...
...
9999 REM CTRL-STOP SUBROUTINE
10000 RETURN
```

REGEL 10	STOP-subroutine wordt gezet op regel 10000
REGEL 20	Schakelt de STOP-detector aan
REGEL 10000	Keert terug naar de plaats waar vandaan de routine werd aangeroepen

Let er op dat ON STOP-detectie de STOP-toets er niet van weerhoudt het programma te laten stoppen. Het voorkomt alleen dat u het programma onderbreekt via <CTRL> <STOP>.
Als u alleen <STOP> indrukt, zult u zien dat het programma stopt en de cursor tevoorschijn komt. Drukt u voor de tweede keer op de STOP-toets, dan gaat het programma gewoon verder.
De <CTRL> <STOP>-interruptie wordt gede-activeerd als het programma niet

meer actief is en ook tijdens fout-detectie-routines.

4. JOYstick-trekkerinterruptie

Gebruikte opdrachten:

ON STRIG GOSUB <een lijst met regelnummers>
STRIG(<n>) ON/OFF/STOP

Het trekker-getal, <n>, is voor iedere trekker als volgt:

0	=	spatiebalk
1	=	trekker 1 van joystick 1
2	=	trekker 1 van joystick 2
3	=	trekker 2 van joystick 1
4	=	trekker 2 van joystick 2

Iedere op MSX passende joystick beschikt over twee trekker-knoppen. De spatiebalk wordt ook beschouwd als trekker.
ON STRIG GOSUB definieert de trekker-interruptie-subroutines voor ieder van deze trekkers. U kunt subroutines definiëren voor alle trekkers tegelijkertijd door het opsommen van alle regelnummers van de trekkers.
Als er geen subroutine bestaat voor een trekker, dan kunt u het regelnummer overslaan en alleen een komma plaatsen.
Bijvoorbeeld:

```
ON STRIG GOSUB 1000,200,,,
```

zal onderbreken voor een subroutine indien:

a. trekker 0 spatiebalk

of:

b. trekker 1 van joystick 1

wordt ingedrukt, maar voert niets uit indien één van de andere trekkers wordt ingedrukt.
STRIG(<n>) ON activeert de trekker-detectie voor de opgegeven trekker. U kunt slechts één trekker tegelijkertijd activeren.
Zodra de interruptie optreedt, wordt er een automatische STRIG(<n>) STOP uitgevoerd. Dit voorkomt dat er een interruptie optreedt gedurende de uitvoering van de huidige trekker-subroutine. Er wordt echter wel onthouden dat een trekker werd ingedrukt, de computer zal onmiddellijk naar de bijbehorende subroutine springen, na het verlaten van de huidige subroutine, tenzij de huidige subroutine de trekker-interruptie de-activeert via STRIG(<n>) OFF. Na het verlaten van de trekker-subroutine voert de computer automatisch een STRIG<(n>) ON uit om de interrupties te activeren.

De STRIG-interruptie wordt uitgeschakeld als het programma niet meer actief is, of als er foutafhandeling plaatsvindt.

Voorbeelden:
Hier volgt een zeer kort programma dat illustreert hoe de trekker-detectie werkt, door het testen van elk van de joystick-trekkers. Het programma is zo geschreven, dat beide joysticks kunnen worden getest, evenals de spatiebalk. Als u de <s>-toets indrukt, stopt het programma.

```
10 ON STRIG GOSUB 100,120,140,160,180
20 STRIG(0) ON
30 STRIG(1) ON
40 STRIG(2) ON
50 STRIG(3) ON
60 STRIG(4) ON
70 IF INKEY$="s" THEN END
80 GOTO 70
90 REM SUBROUTINES
100 PRINT "SPATIEBALK INGEDRUKT"
110 RETURN
120 PRINT "TREKKER 1, JOYSTICK 1"
130 RETURN
140 PRINT "TREKKER 1, JOYSTICK 2"
150 RETURN
160 PRINT "TREKKER 2, JOYSTICK 1"
170 RETURN
180 PRINT "TREKKER 2, JOYSTICK 2"
190 RETURN
```

REGEL 10	Definieert de trekker-subroutines
REGEL 20	Schakelt trekker 0 aan
REGEL 30	Schakelt trekker 1 aan
REGEL 40	Schakelt trekker 2 aan
REGEL 50	Schakelt trekker 3 aan
REGEL 60	Schakelt trekker 4 aan
REGEL 70	Stoppen als <s>-toets wordt ingedrukt
REGEL 80	Springt terug naar regel 70
REGEL 90	Trekker-routines

5. Sprite-botsing-interruptie

Zie hiervoor de geavanceerde graphics sprite-sectie voor een gedetailleerde uitleg.

6. Foutdetectie

Zie hiertoe de sectie foutafhandeling voor meer details.

Hoofdstuk 10

Foutafhandeling

Foutboodschappen

Zodra MSX-BASIC een fout tegenkomt gedurende de uitvoering van een programma of commando, toont hij een foutboodschap en stopt de uitvoering. Hier volgt een overzicht van de foutboodschappen.

CODE	BERICHT	OMSCHRIJVING
1	NEXT without FOR	Een variabele in een NEXT-opdracht is niet te vinden in een vorige FOR-opdracht; of een NEXT-opdracht wordt uitgevoerd zonder dat er een FOR-opdracht aan vooraf ging.
2	Syntax error	Spelfouten in een sleutelwoord of foutieve interpunctie.
3	RETURN without GOSUB	Een RETURN-opdracht zonder een bijbehorende GOSUB-opdracht.
4	Out of DATA	Er wordt een READ gedaan terwijl alle DATA-gegevens reeds zijn ingelezen; het kan ook zijn dat een READ wordt uitgevoerd zonder dat er DATA-opdrachten bestaan.
5	Illegal function call	Er wordt een niet-toegestane parameter opgegeven aan een wiskundige functie of een string-functie.
6	Overflow	Een getal blijkt te groot te zijn voor dit type variabele, functie of opdracht.
7	Out of memory	Er blijft geen geheugenruimte over vanwege een te groot

programma, of te veel FOR's of te veel GOSUB's of te veel functies resp. variabelen.

8 Undefined line number
Er wordt naar een niet aanwezig regelnummer verwezen, vanuit GOTO, GOSUB, IF...THEN...ELSE of vanuit DELETE.

9 Subscript out of range
De beschrijving (lay-out) van een array-element valt buiten de grenzen zoals opgegeven in de DIM-opdracht; ook het getal van de beschrijving kan fout zijn.

10 Redimensioned array
DIM voor een reeds bestaande array.

11 Division by zero
Er blijkt een deling plaats te vinden door nul.

12 Illegal direct
Een opdracht die niet is toegestaan in de directe mode.

13 Type mismatch
Een stringvariabele wordt toegekend aan een getal of omgekeerd.

14 Out of string space
Geheugengebied voor strings opgebruikt in de systeem-RAM.

15 String too long
Een poging tot aanmaak van een string met meer dan 255 tekens lengte.

16 String formula too complex
Een stringuitdrukking blijkt te complex te zijn voor de computer.

17 Can't continue
Een CONTinue wordt gedaan, terwijl het programma met een fout eindigde, gewoon is beëindigd of na een bewerking.

18 Undefined user function
De aanroep van een FN-functie, zonder dat er daarbij een DEF FN aan vooraf ging.

19 Device I/O error
Een in/uitvoerfout met cassette, printer enz. Dit is een fatale fout, zonder een mogelijkheid tot herstel.

20 Verify error
CLOAD? cassettetape verificatie resulteerde in een fout.

21 No RESUME
Een foutdetectie-routine heeft geen RESUME-opdracht, terwijl er wel een door de computer werd verwacht.

22 RESUME without ERROR
Een onverwachte RESUME-opdracht zonder dat er een fout-subroutine werd uitgevoerd.

23 Unprintable error
De fout is onderkend, maar er is geen boodschap.

24 Missing operand
Een uitdrukking bevat een bewerker zonder dat er een operand op volgt.

25 Line buffer overflow
Een ingebrachte regel bevat te veel tekens.

26-49 Unprintable error
Gereserveerd voor toekomstig gebruik.

50 Field overflow
Een FIELD-opdracht probeert meer bytes te reserveren dan waren aangevraagd bij de recordlengte van een willekeurig bestand in de OPEN-opdracht; of het einde van de FIELD-buffer is bereikt, terwijl er sequentiële I/O (PRINT #, INPUT #) werd gedaan naar een willekeurig bestand.

51 Internal error
Fout van de machine.

52 Bad file number
Een opdracht of commando refererend aan een niet-bestaand of buiten de MAXFILES-reeks vallend bestand.

53 File not found
Een LOAD- of OPEN-opdracht refererend aan een niet-bestaand bestand.

54 File already open
Er wordt een poging gedaan om een bestand te openen dat reeds geopend is.

55 Input past end
Er wordt een INPUT-opdracht uitgevoerd terwijl alle gege-

vens in het bestand zijn ingelezen. Gebruik EOF om dit te voorkomen.

56 Bad file name
Een foutieve bestandsnaam werd meegegeven in LOAD, SAVE of een andere I/O-opdracht.

57 Direct statement in file
Er wordt een directe opdracht gegeven terwijl er een bestand wordt geladen (LOAD). Het laden (LOAD) wordt gestopt.

58 Sequential I/O only
Er wordt een opdracht uitgevoerd voor een willekeurig bestand, terwijl er sprake is van een sequentieel bestand.

59 File NOT open
Het gerefereerde bestand blijkt niet open te zijn.

60-255 Unprintable error
Geen foutcodes. Deze mogen door de gebruiker zelf worden gedefinieerd.

Foutafhandeling

De MSX-BASIC bevat vele functies en opdrachten die u behulpzaam kunnen zijn tijdens het oplossen van problemen en het herstellen van fouten. De alledaagse praktijk leert dat u er goed aan doet wanneer u foutafhandeling of foutdetectie-routines in te ontwikkelen programma's opneemt. Dit helpt namelijk bij het localiseren van fouten tijdens het proefdraaien van programma's.
Voordat we verder gaan met de uitleg hoe dergelijke routines worden geschreven, kijken we even naar de BASIC-sleutelwoorden die met foutafhandeling van doen hebben.

ERL staat voor foutregel.
ERL is een gereserveerde variabele die het regelnummer van de regel met de huidige fout bevat.

ERR staat voor foutcode.
ERR bevat de foutcode nadat de computer een fout heeft geconstateerd in een programma. Dit is een waarde die ligt tussen 1 en 255. De betekenis van elk van deze foutcodes kunt u vinden op de vorige pagina's.

FOUT

In principe bestaan er twee manieren waarop de ERROR-opdracht kan worden

gebruikt.

1. Het simuleren van een fout. Een ERROR-opdracht met een integer argument zal de computer misleiden, door hem te laten denken dat er sprake van een fout is, zodat het programma wordt gestopt en daarna de foutboodschap en het regelnummer zullen worden afgedrukt.

2. Om door de gebruiker zelf gedefinieerde fouten te creëren. Een ERROR-opdracht samen met het gebruik van een ON ERROR GOTO foutdetectiefunctie stellen u in staat om uw eigen fouten te creëren (hierover later meer).

ON ERROR GOTO <regel>

ON ERROR GOTO maakt foutdetectie mogelijk en specificeert naar welke regel dient te worden gesprongen in geval van een fout. Zodra de computer is verteld een fout te detecteren, springt hij naar de opgegeven regel, om de foutafhandelingsroutine uit te voeren, indien fouten optreden gedurende een programma of in de directe mode, dus buiten een programma.

RESUME

RESUME staat hier voor het opnieuw starten van een BASIC-uitvoering nadat een foutafhandelingsprocedure plaatsvond, met behulp van de foutdetectie-opdracht ON ERROR GOTO. Nadat de fout is afgehandeld, geeft RESUME de computer aan dat de uitvoering moet worden voortgezet volgens de onderstaande syntax:

RESUME of RESUME 0
Start met de uitvoering vanaf de opdracht die de fout veroorzaakte.

RESUME NEXT
Start met de uitvoering vanaf de opdracht die volgt op de opdracht die de fout veroorzaakte.

RESUME <regel>
Start opnieuw met de uitvoering vanaf de aangegeven regel.

Foutafhandelings-routines

Om een foutafhandelingsroutine in uw programma op te nemen, dient u over het volgende te beschikken:

1. ON ERROR GOSUB-opdracht meteen aan het begin van uw programma, om de foutdetectie te activeren vanaf het allereerste moment. Na de uitvoering van deze opdracht zal de computer waakzaam zijn voor fouten en een omleiding veroorzaken naar de foutafhandelingsroutine in de directe of indirecte mode.

2. Een foutafhandelingsroutine die zich helemaal aan het einde van het programma bevindt.

Een eenvoudig programma met foutafhandeling ziet er als volgt uit:

```
10 ON ERROR GOTO 100
20 PRINT 10
30 PRONT 20
40 PRINT 30
50 END
99 REM FOUTAFHANDELINGSROUTINE
100 ON ERROR GOSUB 0
```

REGEL 10	Activeert de foutdetectie en zet de foutsubroutine op regel 100.
REGEL 30	De fout.
REGEL 50	Einde van het programma.
REGEL 99	ON ERROR GOSUB 0 laat de BASIC stoppen en drukt de fout af, die de detectie veroorzaakte.

Als het bovenstaande wordt uitgevoerd, gebeurt het volgende:

```
RUN
10
Syntax error in 30
```

De fout in regel 30 is gedetecteerd en het programma wordt naar regel 100 gestuurd. ON GOSUB 0 laat de fout zien die verantwoordelijk is voor de detectie en stopt het programma. Ook wordt hierdoor de foutdetectie gedeactiveerd.

Tot nu toe zou ook een foutafhandeling kunnen plaatsvinden zonder de foutdetectie. Zou er geen ON ERROR GOSUB-opdracht zijn geweest, dan zou de computer toch nog steeds de fout in regel 30 detecteren en de boodschap 'Syntax error in 30' geven. De foutdetectie-routine beschikt echter over andere mogelijkheden.

De foutafhandeling zou bijvoorbeeld een herstelprocedure kunnen bevatten, zodat u weer in uw programma terecht zou komen. Om dit te bereiken dient u te weten met welke fouten de herstelroutine overweg dient te kunnen en hoe de computer weer opnieuw moet starten via de RESUME-uitvoering.

In dit voorbeeld blijken er te weinig DATA-elementen te staan in de DATA-opdracht van regel 60. Dit resulteert in een foutmelding 'Out of DATA' (code 4) tijdens de vierde lus. De foutafhandeling is voorzien van een herstelprocedure voor dergelijke fouten in regel 100. Deze regel onderzoekt of de fout inderdaad de 'Out of DATA' is, door te kijken of ERR een 4 bevat en zal indien dit het geval is, een RESTORE uitvoeren opdat de DATA

opnieuw kan worden gebruikt. De uitvoering wordt opnieuw begonnen vanaf de regel waar de fout optrad.

```
10 ON ERROR GOTO 100
20 FOR I=1 TO 5
30 READ A$
40 PRINT A$
50 NEXT I
60 DATA EEN,TWEE,DRIE
70 END
90 REM FOUTROUTINE:
100 IF ERR=4 THEN RESTORE:PRINT 'Out of DATA':RESUME
110 ON ERROR GOTO 0
```

REGEL 10	Activeert de foutdetectie en zet de foutsubroutine op regel 100.
REGEL 20	Vijf maal een lus.
REGEL 30	Leest alle gegevens.
REGEL 40	Drukt A$ af.
REGEL 50	Volgende lus.
REGEL 60	Alleen de drie DATA-elementen
REGEL 70	Einde.
REGEL 100	Indien ERR gelijk is aan 4 (Out of DATA) doe dan een RESTORE-opdracht en begin opnieuw vanaf het punt van onderbreking.
REGEL 110	Drukt een foutboodschap af indien er sprake is van een heel andere fout.

```
RUN
EEN
TWEE
DRIE
Out of DATA
EEN
TWEE
```

Bij sommige fouten wilt u de uitvoering niet opnieuw met dezelfde regel laten beginnen. Fouten als een onjuiste syntax kunnen nooit door de computer worden hersteld en moeten opnieuw worden bewerkt. Echter, indien u de computer deze regel wilt laten overslaan, en u een foutmelding wilt hebben om het programma tot het einde voort te laten gaan, moet u een RESUME NEXT-opdracht gebruiken. Deze laat de uitvoering opnieuw beginnen vanaf de regel die volgt op de regel met de fout.

```
10 ON ERROR GOTO 100
```

```
20 PRINT 10
30 PRoNT 20
40 PRINT 30
50 END
99 REM FOUTAFHANDELINGSROUTINE
100 PRINT "Foutcode ";ERR;
"in regel";ERL
110 RESUME NEXT
```

REGEL 10	Activeert de foutdetectie en zet de foutsubroutine op regel 100.
REGEL 30	Hier zit de fout.
REGEL 50	Einde programma.
REGEL 100	Toont de foutcode en foutregel.
REGEL 110	Uitvoering begint opnieuw bij de volgende regel.

```
RUN
10
Foutcode 2 in regel 30
30
```

Let er op dat u ook een regelnummer kunt meegeven in de RESUME-opdracht, dus in bovenstaand voorbeeld zou RESUME 40 hetzelfde effect hebben gehad. Het is ook mogelijk om de hele regel te tonen waarin de fout optrad, met behulp van een foutafhandelingsroutine. Het zal veel eenvoudiger zijn om te corrigeren, als de regel met de fout direct wordt getoond nadat de fout is gedetecteerd. Om dit te bereiken moet u de foutafhandeling beëindigen met een LIST. (dus LIST punt) -opdracht. LIST. resulteert in een afdruk van de door de computer als laatst gerefereerde regel.

```
10 ON ERROR GOTO 100
20 PRINT 10
30 PRoNT 20
40 PRINT 30
50 END
99 REM FOUTAFHANDELINGSROUTINE
100 PRINT "Foutcode ";ERR;"in regel";ERL
110 LIST
```

REGEL 10	Activeert de foutdetectie en zet de foutafhandelingsroutine op regel 100.
REGEL 30	Hier zit de fout.
REGEL 50	Einde programma.
REGEL 100	Toont foutcode en foutregel.

REGEL 110 Toont de regel met de fout.

```
RUN
10
Foutcode 2 in regel 30
30 PRoNT 20
```

U kunt vervolgens regel 30 corrigeren:

```
30 PRINT 20
```

Hoe u uw eigen fouten kunt maken

Een ERROR-opdracht samen met de ON ERROR GOTO foutdetectie-functie, stellen u in staat om uw eigen fouten te ontwikkelen.
Er zijn ongeveer 36 fouten die liggen tussen de foutcodes 1 en 60 in de huidige versie van MSX-BASIC. Codenummers 61 tot 255 kunnen door de programmeur worden gebruikt.
Om uw eigen fouten te programmeren, dient u eerst de foutdetectie te activeren door het uitvoeren van een ON ERROR GOTO <regel> aan het begin van uw programma. Als dan halverwege de uitvoering van het programma een conditie optreedt die u een foutdetectie wilt laten veroorzaken, dan bereikt u dit door het volgende te doen:

IF <conditie> THEN ERROR <foutcode>

ON ERROR GOTO zal worden ge-activeerd en de computer zal meteen aanvangen met de uitvoering van de foutdetectieroutine. Binnen in de routine moet zich een regel bevinden als:

IF ERR=<foutcode> THEN PRINT "Foutboodschap"

In het volgende voorbeeld zullen alle commando's die voorzien zijn van 'DOODT' worden behandeld als een nieuwe fout (nr. 255), terwijl zij zullen worden behandeld in de foutdetectie-routine met behulp van de ERR-functie.

```
10 ON ERROR GOTO 100
20 INPUT "MAJESTEIT, WAT IS UW OPDRACHT";A$
30 IF INSTR$(A$,"DOODT") THEN ERROR 255
40 PRINT "OK."
50 END
...
100 IF ERR=255 THEN PRINT "DODEN IS NIET TOEGESTAAN IN
DIT AVONTUUR.":RESUME 20
110 END
```

REGEL 10 Activeert de foutdetectie.
REGEL 20 Invoer.
REGEL 30 Controleert op fout woord.
REGEL 100 Foutboodschap en opnieuw beginnen vanaf 20.
REGEL 110 Stoppen als foutcode niet 255 is.

```
RUN
MAJESTEIT, WAT IS UW OPDRACHT? DOODT HEM
DODEN IS NIET TOEGESTAAN IN DIT AVONTUUR
MAJESTEIT, WAT IS UW OPDRACHT? GOTO EAST
OK
```

Het is een gewoonte geworden in MSX-BASIC, dat wanneer u uw eigen fouten definieert, dit te doen vanaf 255 naar omlaag, zodat u geen problemen krijgt met toekomstige wijzigingen in de foutboodschappen door de fabrikanten van MSX-computers.

Opmerkingen bij de foutafhandeling

De ERROR-opdracht kan ook worden gebruikt om het optreden van een fout te simuleren. Bijvoorbeeld:

```
ERROR 2
```

geeft:

```
Syntax error
```

Als de foutcode geen vooraf gedefinieerde boodschap heeft, zal de computer dit onderkennen en 'Unprintable error' afdrukken.
Er is geen foutdetectie actief gedurende de uitvoering van een foutdetectieroutine. Als er een fout optreedt in de foutroutine, dan wordt een foutboodschap gegeven en het programma gestopt.
Niet alle fouten kunnen worden behandeld door de foutdetectie-routines. Het is daarom aan te raden de foutdetectie-routine te beëindigen met een ON ERROR GOTO 0, die de uitvoering van een BASIC-programma stopt en de foutboodschap toont.
ON ERROR de-activeert alle evenement-behandeling-detecties zoals de ON INTERVAL en ON STRIG.

Hoofdstuk 11

Het bewaren en laden via de cassetterecorder

Inleiding

Er bestaan verschillende methoden om gegevens en programma's via een cassetterecorder te bewaren of te laden.
Deze sectie behandelt het meer verfijnde gebruik van de cassetterecorder.

Opdrachten en functies die horen bij het bewaren en laden van de cassetterecorder

Hier volgt een lijst van opdrachten en functies die horen bij het beeldscherm.
Het bewaren en laden van BASIC-programma's (zie hiervoor de inleiding van MSX-BASIC voor meer details).

CLOAD	Laadt een BASIC-programma van cassette.
CSAVE	Bewaart een BASIC-programma op cassette.
CLOAD?	Verifieert een BASIC-programma op tape tegen een programma in het computergeheugen.
MOTOR	Schakelt de motor van de cassette AAN/UIT.

Het bewaren, laden en samenvoegen met behulp van het ASCII-formaat.

SAVE	Bewaart een BASIC-programma op een opgegeven periferie, in een ASCII-bestand.
LOAD	Laadt een BASIC-programma die in ASCII-formaat staat opgeslagen, van een opgegeven periferie.
MERGE	Voegt een BASIC-programma in ASCII-formaat vanuit een bestand toe aan een BASIC-programma in het geheugen van de computer.

Het bewaren en laden van een gedeelte van het computergeheugen als machinecodeprogramma of gegevens.

BSAVE	Bewaart een gedeelte van het geheugen.

BLOAD Laadt een gedeelte van het geheugen.

Het definiëren van de BAUD-snelheid voor het wegschrijven naar cassette.

SCREEN
CSAVE

De BAUD-snelheid van het schrijven naar cassette

De BAUD-snelheid geeft de snelheid aan waarmee gegevens naar een cassettetape worden getransporteerd. De MSX gebruikt het FSK-formaat, dat u de mogelijkheid van twee BAUD-snelheden geeft: 1200 BAUD of 2400 BAUD. Deze kunt u opgeven via de SCREEN-opdracht of het CSAVE-commando. De vervangingswaarde voor het aantal BAUD is 1200. De syntax van deze opdrachten is als volgt:

SCREEN,,,1	1200 BAUD
SCREEN,,,2	2400 BAUD
CSAVE"<naam>"	1200 BAUD
CSAVE"<naam>",1	1200 BAUD
CSAVE"<naam>",2	2400 BAUD

De opneemsnelheid kan zowel 1200 BAUD als 2400 BAUD zijn, maar CLOAD, LOAD en BLOAD zullen automatisch bepalen welke snelheid van toepassing is, en overeenkomstig laden.

Bewaren en laden in het ASCII-formaat

Als een gewoon BASIC-programma op cassette is opgenomen, staat dit opgeslagen in een symbolisch formaat, opdat het programma op een efficiënte manier wordt bewaard. Echter, als u uw programma wilt opnemen met ASCII-codes, dan moet u het SAVE-commando gebruiken. Om een programma te laden dat staat opgeslagen in ASCII-formaat, moet u het LOAD-commando gebruiken. Bij zowel SAVE als LOAD, moet u opgeven welke periferie van toepassing is waarnaar u wegschrijft of waarvan u laadt, in een formaat als <periferie-beschrijving>. Echter, in de huidige MSX-BASIC is alleen de cassette-mogelijkheid van toepassing, zodat de <periferie-beschrijving> altijd 'CAS:' moet zijn. Hier volgt een lijst met de syntax:

SAVE"<periferie-beschrijving> <programmanaam>"
Slaat een BASIC-programma op in ASCII-formaat.

LOAD "<naam van periferie>"
Laadt het eerstvolgende BASIC-bestand, dat stond opgeslagen in ASCII-formaat.

LOAD"<naam van periferie> <bestandsnaam>"
Laadt een BASIC-programma met de opgegeven naam.

LOAD"<naam van periferie> <bestandsnaam>",R
Laadt en voert een opgegeven naam uit.

Let er op dat de R-mogelijkheid het programma automatisch uitvoert nadat het is geladen.
Het belangrijkste gebruik van het bewaren van ASCII-code is, dat het programma kan worden samengevoegd met een ander programma, waar we vervolgens naar zullen kijken.

Het samenvoegen van twee BASIC-programma's

Laten we aannemen dat we twee BASIC-programma's hebben, programma 1 en programma 2, en dat u programma 2 wilt toevoegen aan programma 1, zodat programma 2 volgt op programma 1 als het samenvoegen plaatsvindt. Laten we aannemen dat beide programma's op cassette staan opgeslagen.

PROG 1

```
10 REM PROGRAMMA 1
20 PRINT "WELKOM"
30 PRINT "BIJ"
40 PRINT "MSX"
```

PROG 2

```
10 REM PROGRAMMA 2
20 FOR I=32 TO 58
30 PRINT CHR$(I)
40 NEXT I
```

Als eerste moet u programma 2 laden en dit opnieuw nummeren, opdat de regelnummers groter worden dan die in programma 1.

```
CLOAD "PROG2"
RENUM 50
LIST
```

```
50 REM PROGRAMMA 2
60 FOR I=32 TO 58
70 PRINT CHR$(I)
80 NEXT I
```

Bewaar het programma in een ASCII-bestand met SAVE:

```
SAVE "CAS:PROG2"
```

Laad PROGRAMMA 1 van de cassette:

```
CLOAD "PROG1"
```

Nadat u programma 1 van de cassette heeft geladen, spoelt u de tape terug tot waar programma 2 staat opgeslagen. Tik dan in:

```
MERGE "CAS:PROG2"
```

en start de cassetterecorder. De computer zal nu programma 2 toevoegen aan programma 1.

```
10 REM PROGRAMMA 1
20 PRINT "WELKOM"
30 PRINT "BIJ"
40 PRINT "MSX"
50 REM PROGRAMMA 2
60 FOR I=32 TO 58
70 PRINT CHR$(I)
80 NEXT I
```

Let vooral op het volgende, als u twee programma's samenvoegt:

1. Als een regelnummer zowel in PROGRAMMA 1 voorkomt als in PROGRAMMA 2, dan bevat het uiteindelijke programma de regel van het tweede programma.
2. MERGE heeft de naam van de periferie nodig in de syntax.

   MERGE "<naam van periferie> <bestandsnaam>"
   <naam van periferie>=CAS: voor cassette.

3. Als de bestandsnaam wordt overgeslagen in MERGE, wordt het eerstvolgende ASCII-bestand op tape samengevoegd.

Het bewaren en laden van een gedeelte van het computergeheugen

Gebruik het BSAVE-commando om vanaf een bepaalde geheugenpositie een gedeelte van het computergeheugen te bewaren. U dient zowel start- als eindadres op te geven van het geheugengebied dat u wilt bewaren. Dit commando wordt vooral gebruikt om machinecodeprogramma's en gegevens te bewaren in de vorm van bytes.
Wanneer u een machinecodeprogramma bewaart, heeft u de mogelijkheid om een adres mee te geven waar de uitvoering aanvangt. Als u dan de machinecode

weer laadt met BLOAD, dan start het machinecodeprogramma automatisch met de uitvoering, vanaf het adres dat was opgegeven in BSAVE. Alles dat wordt bewaard met BSAVE moet weer worden geladen met BLOAD.
Als het adres waar de uitvoering moet aanvangen niet is opgegeven, veronderstelt de computer dat hij met de uitvoering moet beginnen vanaf het startadres, als het programma wordt geladen met de R-mogelijkheid om programma's automatisch uit te voeren.
Hier volgt een lijst met de syntax:

BSAVE"<naam van periferie>;<naam>",<startadres>,<eindadres>
BSAVE"<naam van periferie>;<naam>",<startadres>,<eindadres>,
<auto-startadres>
BLOAD "<naam van periferie>;<naam>",R
<naam> en R als mogelijkheid

De R-mogelijkheid start automatisch de uitvoering van de machinecode die zojuist is geladen. R staat voor RUN.

BLOAD "<naam van periferie>;<naam>",<num-const>

Indien de off-set <num-const> wordt meegegeven, wordt het te laden bestand geplaatst op de positie aangegeven door <num-const>.

<naam van periferie>=CAS: voor cassette

De adressen kunnen worden aangegeven in hexadecimale getallen, bijvoorbeeld:

```
BSAVE "CAS:PROG",&HF300,&HF380,&HF30A
```

Hoofdstuk 12

Geavanceerde grafische toepassingen I

Karakteristieken van ieder SCREEN

MSX beschikt over één van de meest veelzijdige grafische chips, de TMS 9929A, ontwikkeld door Texas Instruments Inc. in de VS. Het is één van de meest uitgebreide grafische chips die er momenteel zijn en deze chip biedt u een scala aan mogelijkheden, zoals 16 kleuren met hoog oplossend vermogen en sprite-animatie. De chip zorgt ook voor zijn eigen 16K RAM, hetgeen betekent dat ze de centrale besturing ontlast van het leveren van geheugencapaciteit voor het grafische scherm.
MSX-BASIC is speciaal voorzien van deze krachtige grafische mogelijkheden en eenvoudige programmeerwijze ten behoeve van beginners. Het gebruik blijkt eenvoudig te zijn nadat u er mee heeft geëxperimenteerd en met voldoende inzet enkele spectaculaire plaatjes heeft gemaakt.
Laten we maar eens beginnen met eerst uit te leggen welke schermmodes beschikbaar zijn op de MSX:

SCREENMODE

MODE	TEKST/GRAF.	OPL.VERM.	KLEUR	INPUT	GRAFISCH	SPRITE
0	40x24 tekst	40x24 tekens	2 uit 16	ja	nee	nee
1	32x24 tekst	32x24 tekens	2 uit 16	ja	nee	ja
2	HIRES graph.	256x192 tekens	16	nee	ja	ja
3	Meerkleuren	64x48 blok	16	nee	ja	ja

Opmerkingen:
INPUT - wel of niet gebruik mogelijk van INPUT-opdracht in de gegeven SCREENMODE. U kunt dus niets invoeren in een grafische mode.
GRAFISCH - wel of niet gebruik mogelijk van grafische opdrachten zoals DRAW, in de gegeven SCREENMODE. In het algemeen zijn grafische opdrachten niet mogelijk in de tekstmodes (dus 0 en 1).

Opdrachten en functies die horen bij de SCREENMODES

Hier volgt een lijst van opdrachten en functies die gerelateerd zijn aan het beeldscherm. Meer details vindt u in de BASIC-verwijzingssectie.

Opdrachten behorend bij alle SCREENMODES:

SCREEN	Definieert de SCREENMODE.
CLS	Maakt het scherm schoon.
COLOR	Zet voorgrond-, achtergrond- en randkleur.

Opdrachten en functies behorend bij de tekst-modes:

WIDTH	Zet de breedte van het tekstscherm.
LOCATE	Zet de positie van de cursor op het tekstscherm.
TAB	Zet de horizontale positie van de cursor in de huidige regel.
CSRLIN	Geeft de huidige verticale positie van de cursor.
POS(0)	Geeft de huidige horizontale positie van de cursor.

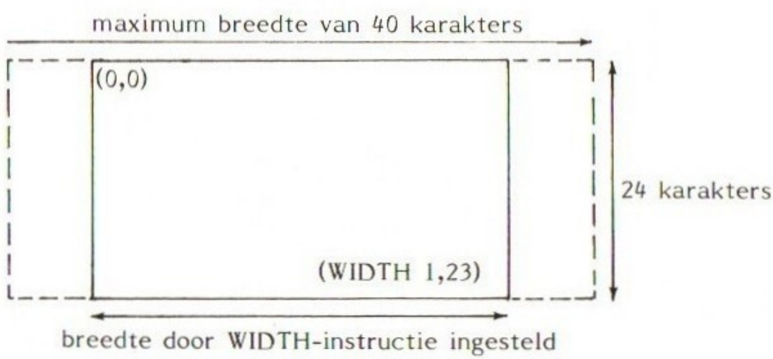
Opdrachten en functies behorend bij grafische modes:

CIRCLE	Trekt een cirkel.
DRAW	Tekent volgens de grafische macrotaal.
LINE	Tekent lijnen, vierkanten en dozen.
PAINT	Kleurt in met de huidige voorgrondkleur.
PSET	Zet een punt.
PRESET	Wist een punt.
POINT	Geeft de kleur van de opgegeven pixel.

Opdrachten behorend bij de VIDEO DISPLAY PROCESSOR:

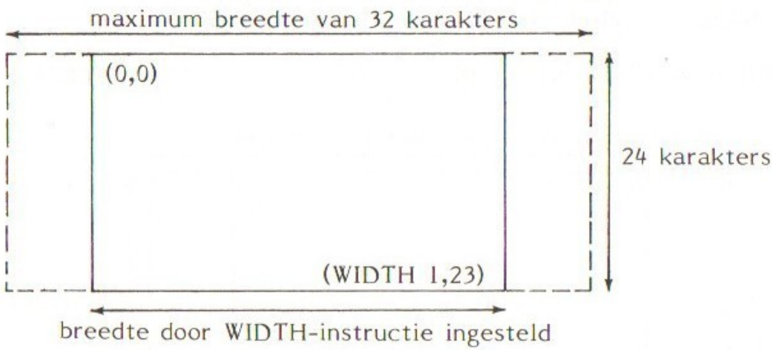
VPOKE	POKE in VIDEO RAM.
VPEEK	PEEK vanuit VIDEO RAM.
VDP	Geeft de waarde van het VDP-register.
BASE	Geeft beginadres van de VIDEO RAM-tabellen.

MODE 0: 40 x 24 TEKSTMODE SCREEN 0



MODE 0 biedt 40 tekens per regel, hetgeen het maximum aantal tekens per regel is dat u kunt gebruiken met MSX-computers. In deze mode kunt u uw programma's schrijven en bewerken. U zult merken dat een weergave van een programma in deze mode eenvoudiger te lezen is.
MODE 0 heeft echter enkele nadelen. De tekens worden bijvoorbeeld in een gecomprimeerde vorm getoond, namelijk 6 bij 8 pixels in plaats van 8 bij 8 pixels, zodat sommige grafische tekens lijken te zijn afgekapt. De twee meest rechtse pixels van ieder teken worden helemaal niet meer getoond! Dit heeft echter geen gevolgen voor de alfanumerieke tekens.
De vervangingsbreedte van het scherm in MODE 0 is 37 tekens. U kunt de beeldbreedte vergroten tot 40 met behulp van de WIDTH-opdracht. De coördinaten van de linkerbovenhoek zijn (0,0) en van de rechteronderhoek zijn dit (WIDTH-1,23), of indien geïnitialiseerd: (38,23).
U kunt de tekstcursor positioneren met TAB en LOCATE. Om uit te vinden waar de cursor zich bevindt, kunt u de commando's POS(0) en CRSLIN gebruiken.
In MODE 0 kunt u geen sprites gebruiken, en u kunt ook slechts twee van de 16 kleuren benutten, terwijl de combinatiekeuze van de twee kleuren geheel door u zelf kan worden bepaald. De vervangingskleur is gelijk aan die van MODE 1, namelijk een witte voorgrondkleur en een blauwe achtergrondkleur.
Let er op dat MODE 0 geen rand heeft: deze verdwijnt gewoon van het beeld, dus het zetten van de randkleur is zinloos in MODE 0. Let er ook op dat de kleur van het beeld onmiddellijk verandert na uitvoering van COLOR.
In tegenstelling tot de grafische schermen, bent u in deze mode vrij in het gebruik van de INPUT-opdracht en de functietoetslijst wordt op regel 23 getoond, tenzij deze is gede-activeerd door KEY OFF.
Alle grafische opdrachten en functies in deze mode worden behandeld als 'Illegal function call', dus kijk hiervoor uit.

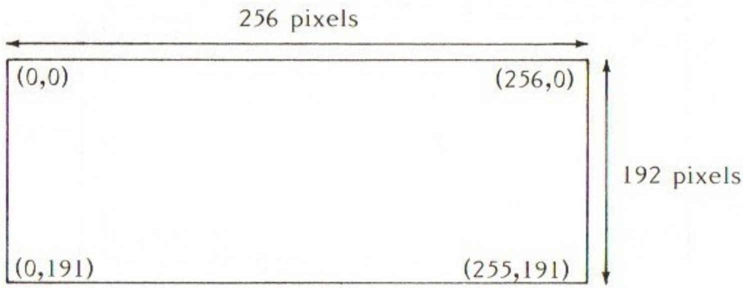
MODE 1: 32 x 24 TEKSTMODE SCREEN 1



MODE 1 heeft een oplossend vermogen van 32 bij 24 tekens maar geen grafische mogelijkheden. In deze mode kunt u programma's schrijven en bewerken. De tekens worden in de afmeting van 8 x 8 pixels getoond zonder afkappingen (zoals het geval was in MODE 0).

De vervangingsbreedte van het scherm is 29 tekens. U kunt de breedte vergroten tot 32 tekens met behulp van de WIDTH-opdracht. De reden van deze kleinere vervangingsbreedte is, dat sommige TV's en monitors niet het volledige scherm kunnen tonen.
De coördinaten van de linkerbovenhoek zijn (0,0) en van de rechteronderhoek (WIDTH-1,23), indien geïnitialiseerd (28,23).
U kunt de tekstcursor positioneren met behulp van TAB en LOCATE. Om uit te vinden waar de cursor zich bevindt, kunt u POS(0) en CRSLIN gebruiken.
U kunt slechts twee van de 16 kleuren gebruiken, terwijl de combinatiekeuze door u zelf kan worden bepaald. Zie hoe de beeldkleur onmiddellijk verandert, na het uitvoeren van de COLOR-opdracht. In tegenstelling tot de grafische modes bent u vrij in het gebruik van INPUT-opdrachten en de functietoetsenlijst wordt getoond op regel 23, tenzij deze is gede-activeerd met KEY OFF.
Alle grafische opdrachten en functies in deze mode, behalve die welke nodig zijn voor sprites, worden als 'Illegal function call' behandeld.

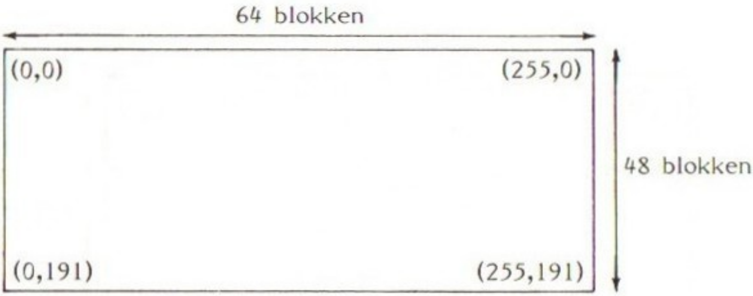
MODE 2: 256 x 192 GRAFISCHE MODE IN HOGE RESOLUTIE
SCREEN 2



MODE 2 is het meest gebruikte grafische scherm, waarbij de gebruiker wordt voorzien van een scherm met een hoog oplossend vermogen (high resolution of HIRES) met 16 kleuren. In deze mode mag u sprites en grafische macrotaal tot DRAW-opdrachten gebruiken. Deze mode wordt vaak bij spellen gebruikt.
De horizontale oplossing is 256 pixels, terwijl de verticale oplossing 192 pixels is. De kleuroplossing echter is iets anders; de kleuroplossing is 32 bij 192 pixels. Dit betekent dat u maar twee kleuren per blok van 8x1 pixels horizontaal kunt kiezen. U kunt de voorgrond- en achtergrondkleur kiezen voor ieder blok van 8 pixels, maar als u een punt afdrukt van een derde kleur, worden alle pixels van de vorige kleur die binnen het blok vallen automatisch veranderd in de nieuwe kleur. Op die manier krijgt u vlekkerige plaatjes als u niet nauwkeurig bent met het afdrukken van pixels en de kleurkeuze. Tekent u echter bedachtzaam, dan kunt u zeer goede

afbeeldingen maken. Later meer over geavanceerde grafische teken-technieken.
U kunt dus sprites gebruiken in deze mode. Sprites worden op een ander schermvlak getoond dan het hoofdschermvlak. Dit betekent dat, welke sprite er ook wordt getoond, niets wordt beïnvloed van wat er zich op het hoofdschermvlak bevindt. U zult zien dat zodra een sprite van het scherm verdwenen is, het hoofdscherm onverstoord is achtergebleven. In de grafische MODE 2 kan de functietoetsenlijst niet worden getoond. U zult ook merken dat de gewone PRINT in deze mode niet werkt. Om op een grafisch scherm iets af te drukken, dient u eerst een bestand te openen naar het scherm toe, en gebruik te maken van PRINT #. Dit wordt later uitvoerig behandeld in de sectie over het afdrukken in de grafische mode.
De COLOR-opdracht zal in deze mode niet direct effect hebben. Om de kleur te veranderen van het hele scherm moet u een CLS laten uitvoeren.
U kunt in deze mode geen INPUT gebruiken, omdat dit het scherm weer in de vorige tekstmode zet.

MODE 3: 64 x 48 MEERKLEUREN GRAFISCH SCHERM MET LAGE RESOLUTIE SCREEN 3



De meerkleuren-MODE 3 biedt een laag oplossend vermogen van 64 bij 48 blokken, maar de individuele blokken kunnen hun eigen kleur hebben (in tegenstelling tot MODE 2). U krijgt dan geen vlekkerig effect zoals in MODE 2, maar dit alles speelt zich dan ook af in een laag oplossend vermogen, zodat er maar een paar toepassingen bestaan voor deze mode.
De coördinaten zijn gelijk aan die van MODE 2, doch een blok wordt hier voorgesteld door 4 x 4 pixels. Dit stelt u in staat om in hetzelfde coördinatenstelsel te tekenen als in MODE 2, hetgeen een belangrijk voordeel is.
U mag sprites gebruiken in deze mode, op eenzelfde wijze als in MODE 2.
In de meerkleuren-MODE 3 kan de functietoetsenlijst niet worden getoond. U zult ook zien dat de gewone PRINT-opdracht niet werkt in deze mode. Om in de grafische mode iets af te drukken, moet u eerst een bestand naar het scherm openen en gebruik maken van PRINT #. De grootte van hetgeen u

afdrukt is vier keer groter dan in de MODE's 1 en 2. Dit zal nader worden uitgelegd in de sectie over het afdrukken in de Grafische Mode.
De COLOR-opdracht zal in deze mode niet direct effect hebben. Om de kleur van het gehele scherm te wijzigen, moet u eerst een CLS-opdracht laten uitvoeren. U kunt INPUT niet gebruiken in deze mode omdat deze opdracht het scherm zal terugplaatsen in de hiervoor gebruikte tekstmode.

Hoofdstuk 13

Geavanceerde grafische toepassingen II

Kleuren in een hoog oplossend vermogen van MODE 2

Dit gedeelte behandelt voornamelijk het gebruik van kleuren in MODE 2. De COLOR-opdracht zet de kleur in de huidige achtergrond, voorgrond en rand. Echter, de achtergrondkleur van het gehele scherm verandert niet eerder dan dat u een CLS laat uitvoeren. Nadat de COLOR is uitgevoerd, heeft iedere punt, elke lijn of elke vorm de nieuwe kleur van de voorgrond aangenomen, tenzij de gebruikte tekenopdrachten anders specificeren.
De kleuroplossing in MODE 2 is 32 x 192 pixels. Dit betekent dat u twee kleuren kunt kiezen per 8 x 1 horizontaal blok pixels. U kunt de voorgrond- en achtergrondkleur opgeven voor ieder blok van 8 pixels, maar als u tracht een punt met een derde kleur te zetten in een blok dat reeds van twee kleuren was voorzien, verandert de oude voorgrondkleur van de pixels automatisch in die van de nieuwe kleur. Dit kan leiden tot vlekkerige resultaten als u niet voldoende oplet. Het volgende korte programma laat zien wat er bedoeld wordt:

```
10 SCREEN 2
20 COLOR 4,15,15
30 CLS
40 IF LINE (0,0)-(4,191),4,BF
50 IF NOT STRIG(0) THEN 50
60 LINE (6,0)-(6,191),10
70 GOTO 70
```

REGEL 10	Kiest MODE 2.
REGEL 20	Zet achtergrondkleur op wit.
REGEL 30	Maakt het scherm schoon en wit.
REGEL 40	Tekent een blauw vierkant.
REGEL 50	Wacht op indrukken spatieblak.
REGEL 60	Tekent een donkergele lijn naast het blauwe vierkant.
REGEL 70	Bevriest het grafische scherm.

Voer dit programma uit. Als eerste ziet u een blauw vierkant verschijnen op een witte achtergrond, aan de rand van het scherm. Als u vervolgens op de spatiebalk drukt, ziet u dat er een gele lijn wordt getrokken en dat nu ook de kleur van het vierkant in geel verandert. Het programma heeft geen geel vierkant getekend op de plaats van het blauwe vierkant. Waarom is dan toch dit vierkant in geel veranderd nadat een gele lijn werd getrokken?
Om te begrijpen wat er is gebeurd, gaan we eens onderzoeken hoe de kleur-attributen staan opgeslagen in video-RAM.
Als u kijkt naar de linker bovenkant van het scherm en dit sterk uitvergroot, dan kunt u dit voorstellen zoals is aangegeven in onderstaande tabel. (B staat voor blauw en W voor wit.)

Y	X=0	1	2	3	4	5	6	7	8	9	.	.	.	.
0	B	B	B	B	W	W	W	W	W	W	W	W	W	W
1	B	B	B	B	W	W	W	W	W	W	W	W	W	W
2	B	B	B	B	W	W	W	W	W	W	W	W	W	W
3	B	B	B	B	W	W	W	W	W	W	W	W	W	W
4														

In de VIDEO-RAM van de MSX wordt bovenstaand plaatje opgeslagen in binaire code, zoals hieronder is aangegeven. Iedere bit stelt een punt (pixel) op het scherm voor, 1=voorgrondkleur en 0=achtergrondkleur. In dit geval staat 1 voor blauw en 0 voor wit.

	0	1	2	3	4	5	6	7	8	9	.	.	.	.
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	0	0	0	0	0	0	0	0	0	0
4														

Deze tabel verraadt achter niet welke kleur iedere punt heeft. Dit komt omdat er een aparte kleurattributentabel staat opgeslagen in de VIDEO-RAM, die de voorgrond- en achtergondkleur definieert van een blok van 8 x 1 pixels. Laten we de linkerbovenhoek nemen en eens kijken hoe deze kleuren in dit geheugen staan opgeslagen.

PATROONTABEL

0	1	2	3	4	5	6	7
1	1	1	1	0	0	0	0

KLEURATTRIBUTENTABEL

1	0
4	15

Zoals u uit bovenstaande tabel kunt afleiden, geeft de patroontabel aan welke pixels in voorgrondkleur en welke in achtergrondkleur staan, terwijl de kleurattributentabel de kleur definieert voor de voorgrond en de achtergrond. Dit betekent dat het fysiek onmogelijk is om meer dan drie kleuren in een blok te hebben. Dus probeert u een gele lijn te trekken in kolom 6, dan wordt allereerst de voorgrondkleur in de attributentabel gewijzigd in de kleurcode die bij geel behoort, dit is 10, en vervolgens wordt pixel 6 op deze nieuwe voorgrondkleur gezet.

PATROONTABEL

0	1	2	3	4	5	6	7
1	1	1	1	0	0	1	0

KLEURATTRIBUTENTABEL

1	0
10	15

De pixels 0, 1, 2 en 3 blijven in de voorgrondkleur, maar de voorgrondkleur zelf is gewijzigd van blauw naar geel en daarmee gepaard gaand is de kleur van de pixels 0, 1, 2 en 3 gewijzigd.
De verklaring is eenvoudig wanneer deze wordt uitgelegd in termen van de geheugentabellen, niet waar?
Dus wat doen we met dit probleem? Er is maar één antwoord: probeer nooit meer dan twee kleuren te gebruiken in een blok.
Bijvoorbeeld bij het voorgaande programma kan dit worden bereikt door de gele lijn niet te tekenen in hetzelfde blok als het blauwe vierkant, doch door het hele plaatje 2 pixels naar links te verplaatsen. Het verbeterde programma ziet er als volgt uit:

```
10 SCREEN 2
20 COLOR 4,15,15
30 CLS
40 LINE (2,0)-(6,191),4,BF
50 IF NOT STRIG(0) THEN 50
60 LINE (8,0)-(8,191),10
70 GOTO 70
```

REGEL 10 Kiest MODE 2.
REGEL 20 Zet witte achtergondkleur.
REGEL 30 Maakt het scherm schoon en wit.

REGEL 40 Tekent een blauw vierkant.
REGEL 50 Wacht op het indrukken van de spatiebalk.
REGEL 60 Tekent een donkergele lijn naast het blauwe vierkant.
REGEL 70 Bevriest het grafische scherm.

Let op de aanpassingen die zijn gemaakt in de coördinaten van regel 40 tot 60.
Wilt u meer weten over de wijze waarop het scherm is opgeslagen in de VIDEO-RAM, kijk dan in de VIDEO-RAM-sectie.

Hoofdstuk 14

Geavanceerde grafische toepassingen III

Hoe op het grafische scherm met behulp van bestanden wordt afgedrukt

Het afdrukken op grafische schermen gaat niet zo recht toe recht aan als afdrukken in de tekstmode. U moet een bestand openen naar het grafische scherm en dan gebruik maken van de PRINT #-opdracht. In principe kunt u ook een bestand openen naar een tekstscherm, maar dit is niet nodig.
Hier volgt een demonstratieprogramma dat naar alle vier de schermen afdrukt, met behulp van bestanden. Regel 30 opent het bestand afhankelijk van welk scherm is gekozen. Er wordt dan een OPEN "CRT:" AS #1 uitgevoerd als geldt dat M kleiner is dan twee. Dit betekent: open een bestand naar een tekstscherm "CRT:" als AS bestandnummer #1. Als geldt dat M groter of gelijk is aan twee, dan wordt uitgevoerd OPEN "GRP:" als (AS) bestandnummer #1. Een geopend bestand dient na uitvoering van de functies te worden gesloten met CLOSE. Dit geschiedt in regel 60: CLOSE#1.

```
10 FOR M=0 TO 3
20 SCREEN M
30 IF M<2 THEN OPEN "CRT:" AS #1
ELSE OPEN "GRP:" AS #1
40 PRINT #1,"Dit is MODE ";M
50 FOR G=1 TO 1000
60 CLOSE #1
70 NEXT M
```

REGEL 10 Lus van 0 tot 3.
REGEL 20 Selecteert schermmode.
REGEL 30 In geval van tekstmode open "CRT:" anders open "GRP:", dit is grafisch scherm.
REGEL 40 Drukt een bericht af.
REGEL 50 Vertraging.
REGEL 60 Sluit een bestand.
REGEL 70 Volgende mode.

In bovenstaand programma ziet u de schermen van mode verwisselen terwijl een bericht 'Dit is MODE ...' wordt afgedrukt. Let er op dat bij een hoog oplossend vermogen de PRINT-opdracht hetzelfde resultaat oplevert als in de tekstmode, dus MODE 1 met 32 x 24 tekens. In de meerkleuren-MODE 3 is de omvang van de tekst viermaal zo groot als die in de gewone tekstmode. Dit komt omdat de meerkleurenmode in een lager oplossend vermogen staat, zodat het afdrukken van tekens moet gebeuren in grafische blokken.

Hoe met een hoog oplossend vermogen in grafische MODE 2 wordt afgedrukt

Als u afdrukt in de grafische MODE 2 kunt u geen gebruik maken van de LOCATE-opdracht om af te drukken tekens te positioneren. Dit komt omdat de computer geen gebruik maakt van een tekstcursor in de grafische mode. Daarvoor in de plaats gebruikt de computer de grafische cursor en drukt daarom af vanaf de laatste positie van deze cursor. Om deze cursor te positioneren kunt u gebruik maken van DRAW, teneinde de grafische cursor te verplaatsen met BM-en (zet spaties) uit de grafische macrotaal.
Voorbeeldprogramma: PRINT # en PRINT # USING in de grafische MODE 2.

```
10 SCREEN 2
20 OPEN "GRP:" AS #1
30 DRAW "BM100,100"
40 PRINT #1,"HELP"
50 DRAW "BM102,102"
60 PRINT #1,"HELP"
70 DRAW "BM100,120"
80 PRINT #1, USING "###.######";ATN(1)*4
90 GOTO 90
```

REGEL 10	Selecteert schermmode 2.
REGEL 20	Opent een bestand naar GRP.
REGEL 30	Positioneert de grafische cursor.
REGEL 40	Drukt tekst af bij cursorpositie.
REGEL 50	Positioneert de cursor opnieuw.
REGEL 60	Drukt opnieuw HELP af.
REGEL 70	Positioneert de cursor opnieuw.
REGEL 80	PRINT # USING.
REGEL 90	Bevriest het grafische scherm.

In bovenstaand voorbeeld worden twee overlappende HELP-berichten getoond. Dit ontstaat doordat in de grafische mode de computer niets uitwist alvorens iets af te drukken. Hiermee kunnen enkele interessante effecten worden bereikt, maar het is vaak ook lastig. De enige manier om een deel van het scherm te wissen bereikt u met behulp van LINE, waarmee wordt overschreven met een vierkant gevuld met de achtergrondkleur.

Als u de volgende regel opneemt:

```
45 LINE (100,100)-STEP (32,8),4,BF
```

wist dit eerst het 'HELP'-bericht zoals gegenereerd in regel 40 en vermijdt aldus het overlappen.
Zoals u in voorgaand programma kunt zien, kunt u ook gebruik maken van PRINT USING in de vorm van PRINT # USING. Alle mogelijke syntaxes van PRINT USING zijn gelijk voor grafische modes en tekstmodes.
Let er op dat ATN(1)*4=PI=3.14159...

Hoe met kleur afdrukken in grafische MODE 2?

Eén van de belangrijkste voordelen van het afdrukken van tekst in de grafische mode is het gebruik van alle 16 kleuren voor letters en vormen. Om met een bepaalde kleur af te drukken behoeft u alleen de voorgrondkleur te specificeren voor de PRINT #.
Hier volgt een voorbeeld om een 16 kleuren-afdruk te illustreren.

```
10 COLOR 1,15,15
20 SCREEN 2
30 OPEN "GRP:" AS #1
40 FOR I=0 TO 15
50 COLOR I
60 DRAW "BM+8,0"
70 PRINT #1,"KLEURCODE";I
80 NEXT I
90 GOTO 90
```

REGEL 10	Achtergrondkleur is wit.
REGEL 20	Schermmode 2.
REGEL 30	Opent bestand naar GRP AS #1.
REGEL 40	LUS.
REGEL 50	Kleur van eerste lus.
REGEL 60	Verplaatst grafische cursor.
REGEL 70	Geeft boodschap.
REGEL 80	Volgende lus.
REGEL 90	Bevriest het scherm.

Dit programma resulteert in het volgende plaatje:

KLEURCODE 0	transparant
KLEURCODE 1	zwart
KLEURCODE 2	middelgroen
KLEURCODE 3	lichtgroen

KLEURCODE 4	donkerblauw
KLEURCODE 5	lichtblauw
KLEURCODE 6	donkerrood
KLEURCODE 7	cyaanblauw
KLEURCODE 8	middelrood
KLEURCODE 9	lichtrood
KLEURCODE 10	donkergeel
KLEURCODE 11	lichtgeel
KLEURCODE 12	donkergroen
KLEURCODE 13	paarsrood
KLEURCODE 14	grijs
KLEURCODE 15	wit

Hoe in meerkleuren-MODE 3 wordt afgedrukt

In deze MODE 3 is de tekst viermaal zo groot als de normale grootte, omdat deze grafische mode een laag oplossend vermogen heeft. Daarom moet voor het afbeelden van tekens gebruik worden gemaakt van grafische blokken, de uiteindelijke tekens zijn dus zeer groot. Hier een demonstratieprogramma:

```
10 SCREEN 3
20 OPEN "GRP:" AS #1
30 DRAW "BM0,0"
40 PRINT#1,"PI"
50 DRAW "BM0,65"
60 PRINT#1,USING"#.######";ATN(1)*4
70 GOTO 70
```

REGEL 10	Selecteert de meerkleurenmode.
REGEL 20	Opent een bestand naar het scherm.
REGEL 30	Positioneert de cursor.
REGEL 40	Drukt 'PI' af.
REGEL 50	Positioneert de cursor opnieuw.
REGEL 60	Drukt de waarde van PI af.
REGEL 70	Bevriest het grafische scherm.

Hoofdstuk 15

Geavanceerde grafische toepassingen IV

Sprites

De MSX-VDP beschikt over de mogelijkheid om sprites te genereren. Sprites zijn door de gebruiker zelf gedefinieerde tekens of vormen, die kunnen worden afgebeeld op het scherm zonder daarbij het oorspronkelijke en onderliggende scherm te verstoren. Dit komt omdat de sprites worden afgebeeld in verschillende schermvlakken. Ze kunnen zo worden gemaakt, dat ze zich achter elkaar kunnen verbergen en bewegen zonder flikkering te veroorzaken. Het biedt u de mogelijkheid om arcadespellen te creëren, zonder daarbij veel problemen te hebben met de grafische voorstellingen.

Grootte van de sprites

Er kan gebruik worden gemaakt van vier maten sprites, doch één sprite-grootte kan tegelijkertijd worden gebruikt. De grootte wordt bepaald met de SCREEN-opdracht.

	GROOTTE	
SCREEN,0	8 bij 8	onvergrote pixel
SCREEN,1	8 bij 8	vergrote pixel 16 x 16
SCREEN,2	16 bij 16	onvergrote pixel
SCREEN,3	16 bij 16	vergrote pixel 32 x 32

Het definiëren van de sprites: SPRITE$

U kunt sprites definiëren met behulp van SPRITE$. U kunt 256 sprite-patronen gebruiken bij de sprite-grootte 0 of 1 (8 bij 8 pixels), of 64 bij sprite-grootte 2 of 3 (16 bij 16 pixels).

Syntax

SPRITE$(<integere waarde>)=<string>

<integere waarde>=sprite-patroonnummer
<integere waarde>=0 tot 255 bij grootte 0 of 1
<integere waarde>=0 tot 63 bij grootte 2 of 3

De SPRITE$ is een string. De lengte van de sprite-string is een vaste lengte van 32 bytes (of tekens), doch voor kleine 8 x 8 sprites behoeft u slechts de eerste acht bytes te definiëren.
Iedere byte in de sprite stelt een rij van acht pixels voor en wordt toegekend aan de sprite-string door het toevoegen van de bijbehorende ASCII-code. Dit wordt bereikt met behulp van de CHR$( )-functie. Bijvoorbeeld om een sprite van 8 x 8 toe te kennen:

SPRITE$(<integer>)=CHR$(<1ste rij>)+CHR$(<2e rij>)
+CHR$(<3de rij>)+CHR$(<4de rij>)+CHR$(<5de rij>)
+CHR$(<6de rij>)+CHR$(<7de rij>)+CHR$(<8ste rij>)

De CHR$( )-functie kan worden vervangen door het eigenlijke teken als u weet welk teken dit is.
Om een sprite van 16 x 16 te definiëren moet u alle 32 bytes opgeven van het sprite-patroon.
Hier volgt een standaardmethode om een sprite van 8 x 8 te definiëren. Teken eerst een sprite op een stukje papier

128	64	32	16	8	4	2	1		BINAIR	DECIMAAL
.	.	.	1	.	.	.	.	=	&B00010000=	16
.	.	1	1	.	.	.	.	=	&B00110000=	48
.	1	1	1	.	.	.	.	=	&B01110000=	112
1	1	1	1	1	1	1	1	=	&B11111111=	255
1	1	1	1	1	1	1	1	=	&B11111111=	255
.	1	1	1	.	.	.	.	=	&B01110000=	112
.	.	1	1	.	.	.	.	=	&B00110000=	48
.	.	.	1	.	.	.	.	=	&B00010000=	16

U kunt dan als volgt een string-uitdrukking afleiden:

```
SPRITE$(0)=CHR$(16)+CHR$(48)+CHR$(112)+CHR$(255)+
CHR$(255)+CHR$(112)+CHR$(48)+CHR$(16)
```

Voorbeeld:
Het gebruik van de binaire getallen is verreweg de eenvoudigste manier om sprites te definiëren, omdat het heel eenvoudig is te bekijken hoe de sprite er uit gaat zien in het programma.

```
10 SCREEN 2,0
20 FOR I=1 TO 8
30 READ A$
40 S$=S$+CHR$(VAL("&B"+A$))
```

```
50 NEXT I
60 SPRITE$(0)=S$
70 PUT SPRITE 0,(100,100),15,0
80 GOTO 80
90 REM BINAIRE VOORSTELLING
100 DATA 00010000
110 DATA 00110000
120 DATA 01110000
130 DATA 11111111
140 DATA 11111111
150 DATA 01110000
160 DATA 00110000
170 DATA 00010000
```

REGEL 10 HIRES-scherm, normale sprite-grootte.
REGEL 30 Leest de binaire getallen als een string.
REGEL 40 S$ is een tijdelijke string.
REGEL 60 Definieert de sprite.
REGEL 70 Plaatst de sprite op positie x=100 en y=100 met een witte kleur in sprite-vlak 0.

Resultaat: u ziet een pijl in het midden van het scherm.
We behandelen de PUT SPRITE-opdracht later nog uitvoeriger.
Als u er niet tegenop ziet om de decimale waarde af te leiden, kan het programma aanzienlijk worden ingekort. Het bovenstaande programma wordt gereduceerd tot de volgende regels:

```
10 SCREEN 2,0
20 SPRITE$(0)=CHR$(16)+CHR$(48)+CHR$(112)+CHR$(255)
+CHR$(255)+CHR$(112)+CHR$(48)+CHR$(16)
70 PUT SPRITE 0,(100,100),15,0
80 GOTO 80
```

REGEL 10 HIRES-scherm, normale sprite-grootte.
REGEL 20 Definieert de sprite 0.
REGEL 70 Plaatst de sprite op positie x=100, y=100, met witte kleur in vlak 0.

Hierna wordt een sprite van 16 bij 16 gedefinieerd:

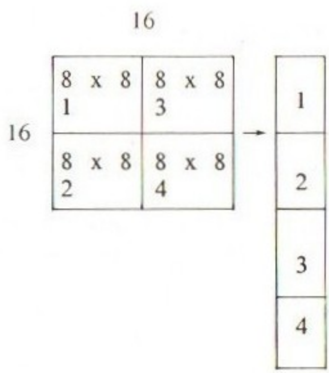
128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
.	.	.	1	1	1	1	1	1	1	1	1	1	.	.	.
.	.	1	1	1	1	1	1	1	1	1	1	1	1	.	.
.	1	1	.	.	.	.	.	.	.	.	.	.	1	1	.
1	1	.	.	.	1	1	1	1	1	1	.	.	.	1	1

```
1 1 . . . 1 1 1 1 1 1 . . . 1 1
1 1 . . 1 . . . . . . 1 . . 1 1
. 1 1 . 1 . . . . . . 1 . 1 1 .
. 1 1 . . 1 . . . . 1 . . 1 1 .
. . 1 1 . . 1 1 1 1 . . 1 1 . .
. . 1 1 . 1 . . . . 1 . 1 1 . .
. . . 1 1 . 1 1 1 1 . 1 1 . . .
. . . 1 1 . . . . . . 1 1 . . .
. . . . 1 1 . . . . 1 1 . . . .
. . . . 1 1 . . . . 1 1 . . . .
. . . . . 1 1 . . 1 1 . . . . .
. . . . . . 1 1 1 1 . . . . . .
. . . . . . . 1 1 . . . . . . .
```

Het bovenstaande diagram in binaire en decimale voorstelling:

00011111	=	31	11111000	=	248
00111111	=	63	11111100	=	252
01100000	=	96	00000110	=	6
11000111	=	199	11100011	=	227
11001000	=	200	00010011	=	19
01101000	=	104	00010110	=	22
01100100	=	100	00100110	=	38
00110011	=	51	11001100	=	204
00110100	=	52	00101100	=	44
00011011	=	27	11011000	=	216
00011000	=	24	00011000	=	24
00001100	=	12	00110000	=	48
00011000	=	12	00110000	=	48
00000110	=	6	01100000	=	96
00000011	=	3	11000000	=	192
00000001	=	1	10000000	=	128

SPRITE-patronen van 16 bij 16 worden op de volgende manier opgeslagen in de video-RAM:



Een sprite van 16 bij 16 = "sprite kwadrant1"+"sprite kwadrant2"+ "sprite kwadrant3"+"sprite kwadrant4".

Hier volgt een redelijk efficiënt programma dat bovengenoemde sprite van 16 bij 16 definieert. De gegevens staan opgeslagen in DATA-opdrachten en worden één voor één toegevoegd aan SPRITE$(0) met de FOR...TO...NEXT-lus.

```
10 SCREEN 2,2
20 FOR I=1 TO 32
30 READ B%
40 S$=S$+CHR$(B%)
50 NEXT I
60 SPRITE$(0)=S$
70 PUT SPRITE 0,(100,100),15,0
80 GOTO 80
90 DATA 31,63,96,199,200,104,100,51
100 DATA 52,27,24,12,12,6,3,1
105 DATA 248,252,6,227,19,22,38,204
110 DATA 44,216,24,48,48,96,192,128
```

REGEL 10	HIRES-schermmode 2, met sprite-grootte 16 bij 16.
REGEL 20	Lus om 32 bytes in te lezen.
REGEL 30	Leest.
REGEL 40	Voegt gegevens toe aan S$.
REGEL 60	Definieert sprite-string 0.
REGEL 70	Plaatst sprite 0 in vlak 0 in wit.
REGEL 90	Gegevens voor kwadrant 1.
REGEL 100	Gegevens voor kwadrant 2.
REGEL 105	Gegevens voor kwadrant 3.
REGEL 110	Gegevens voor kwadrant 4.

Hoe wordt een sprite op het scherm geplaatst: PUT SPRITE

Er zijn 32 sprite-vlakken op het tekst- of grafische vlak, waarbij spritevlak 0 voorop staat. Dit betekent dat hoe kleiner het vlaknummer wordt, hoe hoger de prioriteit is. Als twee sprites elkaar overlappen, zal die met het kleinste vlaknummer vooraan worden afgebeeld en de andere daarachter.
U mag tot en met 256 sprite-patronen definiëren, doch slechts één sprite kan op een specifiek vlak worden getoond. Dit beperkt het maximum aantal sprites dat in één keer kan worden getoond tot 32.
Ook geldt dat u een maximum van vier sprites op een horizontale regel kunt plaatsen. Om een sprite op het scherm af te beelden wordt gebruik gemaakt

van de PUT SPRITE-opdracht. Deze heeft de volgende syntax:

PUT SPRITE <sprite-vlaknummer>[,<coördinaten>]
[,<kleur>][,<patroonnummer>]

Het sprite-vlaknummer mag liggen tussen 0 en 31. De coördinaten geven de positie van de linker bovenhoek van de sprite.
Er zijn twee vormen om coördinaten te specificeren:

1. (<x-coördinaat>,<y-coördinaat>)
   Dit specificeert een absolute positie op het scherm.
2. STEP(<x-verplaatsing>,<y-verplaatsing>)
   Dit bepaalt de relatieve positie ten opzichte van het laatst aangegeven punt.

<x-verplaatsing>,<y-verplaatsing>,<x-coördinaten>,<y-coördinaten> kunnen variabelen, uitdrukkingen of alleen eenvoudige numerieke constanten zijn. Dit betekent dat de sprites kunnen worden bestuurd met behulp van variabelen die u in staat stellen de sprites glijdend te bewegen.
Indien de coördinatenspecificatie wordt weggelaten, plaatst de computer op het laatst een gerefereerde punt.
Voorbeelden:

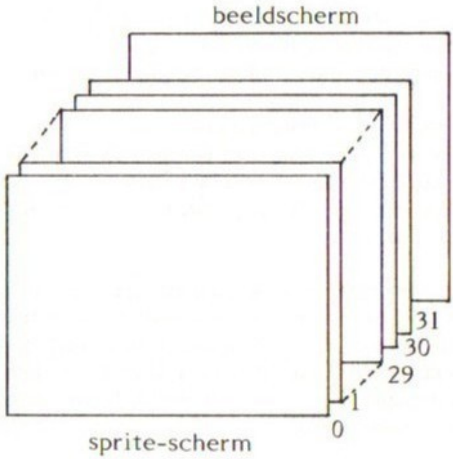
```
PUT SPRITE 0,(50,50),1,1
PUT SPRITE 1,STEP (10,10),12,2
```

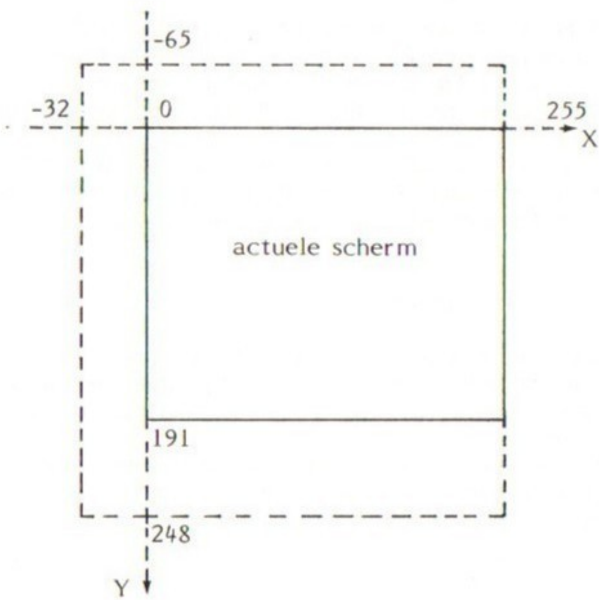
Het sprite-scherm is iets anders dan het eigenlijke beeldscherm. Dit scherm ligt tussen -32 en 255 voor de x-coördinaat en tussen -65 en 248 voor de y-coördinaat. De werkelijke schermcoördinaten zijn kleiner, zodat een sprite die buiten het scherm wordt geplaatst, gedeeltelijke of geheel wordt verborgen.
De sprites zijn voorzien van een volledig scherm-'wraparound'-mogelijkheid. Dit betekent dat, wanneer een sprite buiten het scherm terecht komt, deze terugkeert op de tegenovergestelde zijde van het scherm.

Hoe worden sprites verplaatst?

Door coördinaten te wijzigen in de PUT SPRITE-opdracht, kunt u de sprite naar elke gewenste positie op het scherm verplaatsen. Als de PUT SPRITE-opdracht wordt uitgevoerd, wordt de vorige sprite in de oude positie automatisch volledig uitgewist.
Dit voorbeeldprogramma toont een vierkant dat zich langzaam van rechts naar links verplaatst en aan de linkerzijde buiten het scherm valt, waarna het opnieuw rechts het scherm binnenkomt.

```
10 SCREEN 2,0
20 SPRITE$(0)=STRING$(8,CHR$(255))
```


beeldscherm
31
30
29
1
0
sprite-scherm


-65
-32
0
255
X
actuele scherm
191
248
Y

```
30 FOR X=200 TO -200 STEP -1
40 PUT SPRITE 0,(X,100),1,0
50 FOR D=1 TO 50:NEXT
60 NEXT
70 END
```

REGEL 10	HIRES-scherm, normale grootte.
REGEL 20	Definieert een vierkante sprite.
REGEL 40	Plaatst de sprite op positie X,100.
REGEL 50	Vertraging.

Sprite-kleur

U kunt van elk van de 16 kleuren gebruik maken. Echter, u kunt maar van één kleur gebruik maken binnen een sprite. Het is dus niet mogelijk om meerkleuren-sprites te maken. Om een meerkleuren-sprite te simuleren, kunt u twee of meer sprites van verschillende kleuren op elkaar plaatsen en gezamenlijk verplaatsen.
Let er op dat de achtergrondkleur van een sprite altijd transparant is, zodat u door de gaten in de sprite heen kunt kijken.
Het volgende voorbeeld definieert twee sprites, 0 en 1. Het sprite-nummer 0 wordt in wit afgebeeld en sprite-nummer 1 in donkergeel. Op de afbeelding zult u beide sprites afzonderlijk te zien krijgen, evenals een combinatie van de twee, hetgeen dus een meerkleuren-sprite simuleert.

```
10 SCREEN 2,0
20 SPRITE$(0)=CHR$(16)+CHR$(48)+CHR$(112)+CHR$(255)+CHR
$(255)+CHR$(112)+CHR$(48)+CHR$(16)
30 SPRITE$(1)=CHR$(224)+CHR$(192)+CHR$(128)+CHR$(0)+CHR
$(0)+CHR$(128)+CHR$(192)+CHR$(224)
40 PUT SPRITE 0,(20,20),15,0
50 PUT SPRITE 1,(40,40),10,1
60 PUT SPRITE 2,(60,60),15,0
70 PUT SPRITE 3,(60,60),10,1
80 GOTO 80
```

REGEL 10	HIRES-scherm, normale grootte.
REGEL 20	Definieert sprite 0.
REGEL 30	Definieert sprite 1.
REGEL 40	Plaatst sprite 0 in wit.
REGEL 50	Plaatst sprite 1 in donkergeel.
REGEL 60	Plaatst sprite 2 en 3 op dezelfde coördinaten met twee kleuren, in verschillende schermvlakken.

Hoe sprites kunnen worden verborgen

Als u een speciaal nummer meegeeft aan de y-coördinaat, dan kunt u sprites de-activeren:

Y=208

Als 208 (&HD0) wordt toegekend aan de y-coördinaat, worden alle sprites met een hoger vlaknummer gede-activeerd, totdat een andere waarde dan 208 wordt toegekend aan dat vlak.

```
10 SCREEN 2,0
20 SPRITE$(0)=STRING$(8,CHR$(255))
30 PUT SPRITE 0,(20,20),15,0
40 PUT SPRITE 1,(40,40),15,0
50 PUT SPRITE 2,(60,208),15,0
60 PUT SPRITE 3,(80,80),15,0
70 PUT SPRITE 4,(100,100),15,0
80 GOTO 80
```

REGEL 10	HIRES-scherm, normale grootte.
REGEL 20	Definieert een vierkante sprite.
REGEL 50	Y=208 de-activeert sprite-vlaknummer 2 en verder: sprites van regel 60 en 70 worden niet getoond.

Als 209 (&HD1) wordt toegekend aan Y, verdwijnt die sprite van het scherm.

```
10 SCREEN 2,0
20 SPRITE$(0)=STRING$(8,CHR$(255))
30 PUT SPRITE 0,(20,20),15,0
40 PUT SPRITE 1,(40,40),15,0
50 PUT SPRITE 2,(60,209),15,0
60 PUT SPRITE 3,(80,80),15,0
70 PUT SPRITE 4,(100,100),15,0
80 GOTO 80
```

REGEL 10	HIRES-scherm, normale grootte.
REGEL 20	Definieert een vierkante sprite.
REGEL 30	Vierkante sprite in vlaknummer 1.
REGEL 40	Vierkante sprite in vlaknummer 2.
REGEL 50	Y=209 de-activeert deze sprite.
REGEL 60	Vierkante sprite in vlaknummer 3.
REGEL 70	Vierkante sprite in vlaknummer 4.

De vijfde sprite-regel

Er is een maximum van vier sprites die op een horizontale regel kunnen worden getoond. Wordt deze regel overtreden, dan worden alleen de vier sprites van de laatste vlaknummers normaal getoond. De rest wordt op die regel niet getoond. Het sprite-nummer van de overtredende vijfde sprite kan worden gevonden vanuit het VDP-statusregister met behulp van de VDP-functie (zie het gedeelte over de VDP-functie).
Voorbeeld:

```
10 SCREEN 2,0
20 SPRITE$(0)=STRING$(8,CHR$(255))
30 PUT SPRITE 0,(20,100),15,0
40 PUT SPRITE 1,(40,100),15,0
50 PUT SPRITE 2,(60,100),15,0
60 PUT SPRITE 3,(80,100),15,0
70 PUT SPRITE 4,(100,104),15,0
80 GOTO 80
```

REGEL 10	HIRES-scherm, normale grootte.
REGEL 20	Definieert een vierkante sprite.
REGEL 30	Vierkante sprite op regel 100.
REGEL 40	Vierkante sprite op regel 100.
REGEL 50	Vierkante sprite op regel 100.
REGEL 60	Vierkante sprite op regel 100.
REGEL 70	Vierkante sprite op regel 100.

In bovenstaand voorbeeld ziet u vier vierkanten op dezelfde horizontale regel 100. De vijfde sprite bevindt zich op regel 104 en toont dus de helft van het vierkant, de andere helft blijft verborgen door de vijfde sprite-regel. Zou het regel 108 zijn geweest, dan was de complete sprite zichtbaar geworden.

Animatie met sprites

Eenvoudige sprite-animatie kan zonder al te veel rompslomp worden bereikt op de MSX. Het enige dat u hoeft te doen, is het snel verwisselen van twee sprite-patronen, zodat het lijkt alsof de figuren bewegen.
Hieronder volgt een eenvoudig programma dat een ruimtemonster toont met bewegende poten. De SPRITE$(0) bevat het monster met gesloten poten, SPRITE$(1) dat met open poten.

```
10 SCREEN 2,1
20 SPRITE$(0)=CHR$(60)+CHR$(126)+CHR$(129)+CHR$(219)+CH
R$(126)+CHR$(36)+CHR$(36)+CHR$(36)
```

```
30 SPRITE$(1)=CHR$(60)+CHR$(126)+CHR$(129)+CHR$(219)+CH
R$(126)+CHR$(36)+CHR$(60)+CHR$(129)
40 PUT SPRITE 0,(100,100),11,0
50 FOR I=1 TO 500:NEXT
60 PUT SPRITE 0,(100,100),11,1
70 FOR I=1 TO 500:NEXT
80 GOTO 40
```

REGEL 20	Monster met gesloten poten.
REGEL 30	Monster met open poten.
REGEL 40	Sprite 0 (gesloten poten) geel.
REGEL 50	Vertraging.
REGEL 60	Sprite 1 (open poten) geel.
REGEL 70	Vertraging.

Demonstratieprogramma

In het volgende programma krijgt u twee planeten te zien die in een baan om de zon bewegen. Het gehele stelsel drijft vanaf linksboven op het scherm.

```
10 SCREEN 2,0
20 COLOR 15,1,1
30 CLS
40 SPRITE$(0)=CHR$(126)+STRING$(6,CHR$(255))+CHR$(126)
50 SPRITE$(1)=STRING$(3,CHR$(0))+CHR$(24)+CHR$(24)+STRI
NG$(3,CHR$(0))
60 FOR I=0 TO 6.28 STEP .2
70 X=X+1.5
80 Y=Y+1
90 X1=30*COS(I)
100 Y1=30*SIN(I)
110 X2=15*COS(I)
120 Y2=15*SIN(I)
130 PUT SPRITE 0,(X,Y),11,0
140 PUT SPRITE 1,(X1+X,Y1+Y),9,1
150 PUT SPRITE 2,(X2+X,Y2+Y),15,1
160 NEXT
170 GOTO 60
```

REGEL 10	HIRES-scherm, normale grootte, zwarte rand en achtergrond, witte voorgrond.
REGEL 30	Maakt het scherm schoon.
REGEL 40	Sprite 0 is de zon.

REGEL 50	Sprite 1 is een planeet.
REGEL 60	LUS.
REGEL 70	Verplaatst de zon in X, met 1.5.
REGEL 80	Verplaatst de zon in Y, met 1.
REGEL 90	X1=X-coördinaat voor planeet 1.
REGEL 100	Y1=Y-coördinaat voor planeet 1.
REGEL 110	X2=X-coördinaat voor planeet 2.
REGEL 120	Y2=Y-coördinaat voor planeet 2.
REGEL 130	Plaatst zon in nieuwe positie.
REGEL 140	Plaatst planeet 1 in nieuwe positie.
REGEL 150	Plaatst planeet 2 in nieuwe positie.
REGEL 160	Volgende lus.
REGEL 170	Doorloopt nogmaals de lus.

Hoe een botsing tussen twee sprites wordt gedetecteerd

De TMS 9918/9929 VDP detecteert botsingen tussen sprites. Dit is een nuttige mogelijkheid voor die spelen, waarbij het schieten en ontwijken van de vijand noodzakelijk is. Er zijn twee BASIC-opdrachten die met sprite-botsingen te maken hebben: ON SPRITE GOSUB en SPRITE ON/OFF/STOP.
De ON SPRITE GOSUB-opdracht definieert de sprite-botsing-interruptie-routine. Deze vertelt de computer vanaf welk regelnummer de sprite-botsing-routine begint.
De SPRITE ON-opdracht activeert de eigenlijke botsing-detectie die het programma dirigeert naar de met ON SPRITE GOSUB aangegeven routine indien twee sprites botsen. U moet SPRITE ON uitvoeren alvorens de detectie start.
Zodra een pixel van beide sprites elkaar overlappen, wordt dit gezien als een botsing. Dit treedt ook op als de sprites transparant zouden zijn.
Ook al kan de computer een botsing constateren, hij kan niet aangeven welke sprites hebben gebotst, en ook niet waar de botsing plaatsvond. U dient dit zelf te programmeren.
Zodra een interruptie optreedt, dit is als twee sprites botsen, wordt een automatische SPRITE STOP uitgevoerd. Dit voorkomt dat een tweede aanroep van de subroutine optreedt ten gevolge van een volgende botsing, tijdens de uitvoering van de huidige sprite-botsing-routine. Echter, er wordt wel onthouden dat er een andere botsing is opgetreden tijdens deze routine, zodat direct na het beëindigen van de huidige routine opnieuw deze subroutine wordt uitgevoerd, tenzij dit wordt tegengehouden door een de-activering van de detectie door middel van een SPRITE OFF.
Nadat de sprite-subroutine wordt verlaten, voert de computer automatisch een SPRITE ON uit om de onderbreking weer mogelijk te maken, tenzij een SPRITE OFF werd uitgevoerd in de subroutine.
Na een SPRITE OFF-opdracht stopt de computer volledig met het detecteren van een sprite-botsing.

Voorbeeld:
In het volgende voorbeeld krijgt u twee vierkante sprites te zien, een gele en een witte, die elkaar naderen vanuit de tegenovergestelde richting op

het scherm. Zodra ze botsen, treedt er een ON SPRITE GOSUB op en wordt een sprong uitgevoerd naar de sprite-botsingroutine. De SPRITE OFF in de sprite-interruptieroutine voorkomt dat navolgende botsingen worden gedetecteerd. Dit is gedaan omdat de sprites elkaar blijven overlappen tijdens de uitvoering van de subroutine, maar ook nog erna. Was geen SPRITE OFF uitgevoerd, dan ontstaat een eeuwige lus waarin de sprite-subroutine wordt uitgevoerd. (Probeer de routine maar eens uit te voeren zonder SPRITE OFF, en zie wat er gebeurt.)

```
10 ON SPRITE GOSUB 110
20 SCREEN 2,0
30 SPRITE$(0)=STRING$(8,CHR$(255))
40 SPRITE$(1)=STRING$(8,CHR$(255))
50 SPRITE ON
60 FOR I=10 TO 240
70 PUT SPRITE 0,(I,100),11,0
80 PUT SPRITE 0,(250-I,100),15,0
90 NEXT I
100 GOTO 50
110 REM SPRITE BOTSING
120 SPRITE OFF
130 BEEP
140 RETURN
```

REGEL 10	Zet de detectieroutine op regel 110
REGEL 20	Definieert een grafisch scherm.
REGEL 30	Sprite 0 is een vierkant.
REGEL 40	Hetzelfde geldt voor sprite 1.
REGEL 50	Sprite-botsingdetector-activatie.
REGEL 60	Lus.
REGEL 70	Sprite (geel) verplaatst vanuit links.
REGEL 80	Sprite (wit) verplaatst vanuit rechts.
REGEL 90	Volgende lus.
REGEL 100	Opnieuw starten van de lus.
REGEL 110	De-activeert detectie (één keer is genoeg).
REGEL 120	BEEP-indicatie.
REGEL 130	Ga terug naar de lus.

Opmerkingen bij sprite-botsingen

De computer controleert de sprite-botsingen na iedere uitgevoerde opdracht. Daarom mag er niet van uit worden gegaan dat de computer een sprite-botsing in een bepaalde tijdsinterval constateert.
De VDP-functie geeft u de status van de sprite-botsing-indicator in register 8 van de Video Display Processor (zie het gedeelte over de VDP).

Hoofdstuk 16

Geavanceerde grafische toepassingen V
Hoe men toegang verkrijgt tot de Video Display Processor

VDP-functie

De VDP TMS 9929A beschikt over acht registers die alleen weg kunnen schrijven en één register dat alleen kan lezen. U kunt deze VDP-registers benaderen vanuit BASIC, met behulp van de VDP-functie.
Voor de registers die alleen weg kunnen schrijven, VDP(0) tot VDP(7), geldt dat u alleen waarden aan ze kunt toekennen, dit is VDP(1)=2. Er bestaat geen mogelijkheid om de inhoud van de registers direct te lezen vanuit de VDP.
De computer bewaart echter een kopie van de inhoud van deze registers in het systeem-werkgeheugen. Dit opslaggebied bevindt zich tussen de locaties ROGSAV en RG7SAV. (Zie het gedeelte over de systeem-RAM in de BIOS-verwijzingssectie voor uitvoerige details.)
VDP(8) is het register dat alleen kan lezen. U kunt dit register als volgt lezen:

```
PRINT VDP(8)
```

U moet onthouden dat, indien u een foutieve waarde toekent aan één van deze registers, het scherm opgeknoopt kan worden. Als dit gebeurt dient u de computer uit/aan te schakelen om zodoende het VDP te initialiseren. Wees er zeker van wat u gaat doen voordat u de VDP-functie gebruikt.

VDP-registers

Registers die alleen wegschrijven:

Register 0................VDP(0)

Register 0 bevat twee VDP-besturingsbits, bit 1 (M3) en bit 0 (EV). Bit 2 tot bit 7 moeten 0 zijn.

BIT 0 EV:	externe VDP. Deze activeert/de-activeert invoer vanaf een externe VDP. EV=0 voor de meeste MSX-machines. 0 = de-activeren.

	1 = activeren.
BIT 1 M3:	mode-bit 3. M3 wordt gebruikt om de schermmode te selecteren en wordt samen met register 1 gebruikt. (Zie ook register 1.)

RO

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	0	M3	EV=0

Register 1...............VDP(1)

Register 1 bevat 7 VDP-besturingsbits. Bit 2 is gereserveerd en moet altijd 0 zijn.

BIT 0 MAG: Vergrotingsmogelijkheid voor sprites.
0 = onvergroot 1x1 pixel-oplossing.
1 = vergroot 2x2 pixel-oplossing.

BIT 1 SIZE: Selector voor sprite-grootte.
0 = 8x8 pixel-sprites.
1 = 16x16 pixel-sprites.

BIT 2 Gereserveerd = 0.

BIT 3 M2: Mode-bit 2. M2 wordt gebruikt om de schermmode te selecteren.

BIT 4 M1: Mode-bit 1. M1 wordt gebruikt om de schermmode te selecteren.
Een combinatie van M1, M2 en M3 geeft de uiteindelijke schermmode.

M1	M2	M3	
0	0	0	Tekstmode 1
0	0	1	HIRES, grafische mode 2
0	1	0	Meerkleuren, grafische mode 3
1	0	0	Tekstmode 0

BIT 5 IE: Activeert interruptie.
0 = de-activeert VDP-interruptie.
1 = activeert VDP-interruptie.

BIT 6 BL: Activeert/de-activeert het scherm. Dit biedt u de mogelijkheid om het scherm uit te schakelen.
0 = zorgt dat het scherm wegvalt, maar de randkleur getoond blijft.
1 = activeert het beeldscherm.

BIT 7 VRAM: Selecteert welk soort video-RAM wordt gebruikt. (VRAM=1 bij de meeste MSX-machines.)
0 = 4K RAM.
1 = 16K RAM.

R1

B7	B6	B5	B4	B3	B2	B1	B0
VRAM	BL	IE	M1	M2	0	SIZE	MAG

Register 2...............VDP(2)

Register 2 definieert de startpositie van de namentabel. De waarde varieert van 0 tot 15. Dit register vertegenwoordigt de eerste vier bits van het 14 bits adres van de namentabel. Daarom moet u het adres schrijven gedeeld door &H400.

R2

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	ADRES NAMENTABEL			

NOOT: R2*&H400=adres van namentabel.

Register 3...............VDP(3)

Register 3 definieert het beginadres van de kleurentabel. De waarde ervan kan variëren tussen 0 en 255. Dit register vertegenwoordigt de eerste 8 bits van het 14 bits-adres van de kleurentabel. Daarom schrijft u het adres gedeeld door &H40.

R3

B7	B6	B5	B4	B3	B2	B1	B0
ADRES VAN KLEURENTABEL							

NOOT: R3*&H40=adres van de kleurentabel.

Register 4...............VDP(4)

Register 4 definieert het beginadres van de patroon-, tekst- of meerkleuren-generatortabel, afhankelijk van de schermmode. Zijn waarde kan variëren tussen 0 en 7. Dit register vertegenwoordigt de eerste 3 bits van het 14 bits-adres van de patroon-/tekst-/meerkleurentabel. Daarom moet u het adres schrijven, gedeeld door &H800.

R4

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	PAT GEN ADD		

NOOT: R4*&H800=adres van patroon-/tekst-/meerkleurentabel.

Register 5...............VDP(5)

Register 5 definieert het beginadres van de sprites-attributentabel. Zijn waarde ligt tussen 0 en 127. Dit register vertegenwoordigt de eerste 7 bits van het 14 bits-adres van de sprites-attributentabel. Daarom schrijft u het adres gedeeld door &H80.

R5

B7	B6	B5	B4	B3	B2	B1	B0
0	SPRITES ATTRIBUTEN TABEL ADRES						

Register 6...............VDP(6)

Dit register definieert het beginadres van de sprite-patroongenerator-tabel. Zijn waarde ligt tussen 0 en 127. Dit register vertegenwoordigt de eerste 3 bits van het 14 bits-adres van de sprite-patroongeneratortabel. Daarom schrijft u dit adres gedeeld door &H800.

R6

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	SPR PAT GE.ADR		

NOOT: R6*&H800=adres van de sprite-patroongeneratortabel.

Register 7...............VDP(7)

Register 7 is verdeeld in twee helften. De eerste vier bits stellen de voorgrondkleur voor van de tekstmode, terwijl de laatste vier bits de achtergrondkleur bevatten voor zowel de tekst als de grafische modes.
De waarde van de voorgrondkleur ligt tussen 0 en 15. Hetzelfde geldt voor de achtergrondkleur.

R7=<voorgrondkleur>*&H10+<achtergrondkleur>

R7

B7	B6	B5	B4	B3	B2	B1	B0
VOORGRONDKLEUR				ACHTERGRONDKLEUR			

Het register dat alleen leest

Het status-register...............VDP(8)

De status-indicator bevat de indicatie dat de interruptie actief is, en

een indicatie voor sprite-botsing en de vijfde sprite.

Interruptie-indicatie F: BIT 7

De interruptie-indicator wordt op 1 gezet, nadat het schermraster volledig is afgetast. Het wordt teruggezet op 0 nadat het statusregister is gelezen, of wanneer de VDP extern wordt teruggezet.

Sprite-botsing-indicatie C: BIT 5

De VDP controleert of twee of meer sprites elkaar overlappen met minstens 1 pixel. Dit gebeurt iedere 1/50ste van een seconde.
1 = sprite-botsing.
0 = geen overlapping.

Vijfde sprite-indicator 5S: BIT 6

5S wordt op 1 gezet indien er meer dan vier sprites op een horizontale regel staan (regels 0 tot 192). Let er op dat de VDP met niet meer dan vier sprites overweg kan op een enkele regel. Zijn er vijf of meer sprites, dan wordt de vijfde sprite niet getoond.

Vijfde sprite-nummer

Zijn er meer dan vier sprites op een horizontale regel en staat 5S op 1, dan wordt het sprite-nummer van de overtredende vijfde sprite vervangen door het nummer van de vijfde sprite.

R2

B7	B6	B5	B4	B3	B2	B1	B0
F	5S	C	VIJFDE SPRITENUMMER				

Hoofdstuk 17

Geavanceerde grafische toepassingen VI

De video-RAM

De MSX beschikt over 16 K video-RAM, die gescheiden is van het hoofdgeheugen voor programmeren. Deze wordt bestuurd en ververst door de Video Display Processor en is daarom onafhankelijk van de centrale besturing van de Z80. (Dynamische RAM's die worden gebruikt door MSX moeten worden ververst opdat geen gegevens verloren gaan.)
Dit biedt de volgende voordelen:

1. De Z80 hoeft de RAM voor het beeldscherm niet te verversen, hetgeen tijd bespaart.
2. De grafische beelden gebruiken geen kostbare geheugenruimte van de hoofd-RAM die wordt bestuurd door de Z80.

De video-RAM wordt opgedeeld in diverse sectoren, ieder met een eigen functie.
Het is zeer wel mogelijk om toegang te verkrijgen tot de video-RAM vanuit BASIC met behulp van de volgende opdrachten en functies:

BASE	Geeft het beginadres van de diverse tabellen in de video-RAM.
VPEEK	Leest de inhoud van de video-RAM.
VPOKE	Schrijft gegevens naar de video-RAM.

De BASE-sectie geeft de huidige beginadressen van de verschillende beeldschermtabellen in de video-RAM. (Zie BASE voor meer nauwkeurige details.)
Zie de tabel op de volgende pagina.

NAAMTABEL EN POSITIE IN HEX	TEKST-MODE 0	TEKST-MODE 1	GRAFISCHE MODE 2	GRAFISCHE MODE 3
patroonnamen-tabel	&H0000	&H1800	&H1800	&H0800
kleurentabel		&H2000	&H2000	
patroongenerator-tabel	&H0800	&H0000	&H0000	&H0000
sprite-attribu-tentabel		&H1B00	&H1B00	&H1B00
sprite-genera-tortabel		&H3800	&H3800	&H3800

Hoofdstuk 18

Geavanceerde geluidseffecten met de PSG

De geluidsgeneratorprocessor van de MSX, de AY-3-8912, is al enige tijd in de roulatie. Hij werd ontwikkeld door General Instruments aan het einde van de jaren zeventig en is nog steeds de meest populaire geluids-chip. Hij wordt veel gebruikt in kleine 8 bits-computersystemen en machines met arcadespellen. De reden van het succes ligt hoofdzakelijk in de flexibiliteit ervan en de mogelijkheid om drie kanalen van periodiek geluid en een witte ruis-kanaal tegelijkertijd te genereren. Ook al omdat de chip zelf een microprocessor is, zodat eventueel te genereren geluid geen oponthoud veroorzaakt in de uitvoering van de hoofdbesturing. Ondanks het feit dat de bewuste chip een beetje uit de tijd is geraakt, kunt u erg veel bereiken met deze geluidsgenerator.
Dit gedeelte gaat over de werkzaamheden van de AY-3-8910-geluidsgenerator en het gebruik van de SOUND-opdracht.
Als voorbeeld kunt u het volgende programma eens uitvoeren, dat een vreemd geluid genereert van een vliegende schotel.

```
10 SOUND 0,87
20 SOUND 7,62
30 SOUND 8,16
40 SOUND 11,179
50 SOUND 12,45
60 SOUND 13,14
```

Golfkarakteristieken die door de PSG worden gegenereerd

In principe kan de PSG twee soorten golven genereren: een periodieke vierkantsgolf en witte ruis.
Voorbeeld van een periodieke golf:

```
10 SOUND 7,62
20 SOUND 8,15
30 SOUND 1,1
40 SOUND 0,28
```

Voorbeeld van witte ruis:

```
10 SOUND 7,55
20 SOUND 8,15
```

De geluiden kunnen echter zodanig worden omgevormd in complexe geluiden dat het levensecht klinkt. Om dit te bereiken moet u beschikken over de besturing van de frequentie, geluidssterkte, toon en envelope van ieder geluidskanaal met behulp van de SOUND-opdracht die de PSG binnentreedt.

PSG-geluidsregisters

AY-3-8910 PSG beschikt over 14 registers waarin u kunt schrijven met behulp van SOUND. De syntax van SOUND is als volgt:

SOUND <registernummer>,<waarde om op te slaan>

Door de juiste waarden op te slaan in deze registers kunt u kanalen, geluidsgolven, toonfrequenties, ruis en geluidssterkte selecteren.
Er zijn 14 registers die de volgende functies hebben:

REGISTER	BESCHRIJVING
0	Precieze toonfrequentie van kanaal A.
1	Grove toonfrequentie van kanaal A.
2	Precieze toonfrequentie van kanaal B.
3	Grove toonfrequentie van kanaal B.
4	Precieze toonfrequentie van kanaal C.
5	Grove toonfrequentie van kanaal C.
6	Frequentie van de ruisgenerator.
7	Mixer-schakelaars aan/uit van kanalen A, B en C.
8	Volumeregeling kanaal A.
9	Volumeregeling kanaal B.
10	Volumeregeling kanaal C.
11	Envelope-periode (laag).
12	Envelope-periode (hoog).
13	Patroon-envelope.

Het kiezen van geluid per kanaal: register 7, de mixer

Dit register schakelt het geluid aan/uit voor ieder van de drie kanalen. Het kan de herkomst van de geluidsgolven selecteren vanuit de ruisgenerator of vanuit de individuele toongenerator voor ieder kanaal.
De bits 5, 4 en 3 van register 7 schakelen de ruisgenerator aan voor respectievelijk de C-, B- en A-kanalen. 1 betekent uit en 0 betekent aan. De bits 2, 1 en 0 van register 7 schakelen de toongenerator aan voor respectievelijk de kanalen C, B en A. Dus als voorbeeld: als u kanaal C een

toon wilt laten uitzenden, dan moet bit 2 op 0 worden gezet.

BIT	B7	B6	B5	B4	B3	B2	B1	B0
	/	/	GELUID			TOON		
	/	/	C	B	A	C	B	A
	0	0	1	1	1	0	1	1
			UIT	UIT	UIT	AAN	UIT	UIT

Het formaat van de SOUND-opdracht hierboven is als volgt:

SOUND 7,&B00111011 of decimaal SOUND 7,59

De toon in kanaal A hangt af van register 1 en 0, maar hierover later meer Wilt u echter kanaal B witte ruis laten uitzenden en C en A als hierboven laten staan, dan wijzigt register 7 in:

BIT	B7	B6	B5	B4	B3	B2	B1	B0
	/	/	GELUID			TOON		
	/	/	C	B	A	C	B	A
	0	0	1	0	1	0	1	1
			UIT	AAN	UIT	AAN	UIT	UIT

Het formaat van de SOUND-opdracht hierboven is als volgt:

SOUND 7,&B00101011 of decimaal SOUND 7,43

U kunt een kanaal zowel witte ruis als een toon laten genereren, als u beide aanschakelt. Gesteld, u wilt alle kanalen zowel toon als ruis laten genereren, dan krijgt u:

BIT	B7	B6	B5	B4	B3	B2	B1	B0
	/	/	GELUID			TOON		
	/	/	C	B	A	C	B	A
	0	0	0	0	0	0	0	0
			AAN	AAN	AAN	AAN	AAN	AAN

Het formaat van de SOUND-opdracht op de vorige pagina is als volgt:

SOUND 7,&B00000000 of decimaal SOUND 7,0

NOOT: bits B7 en B6 worden niet gebruikt en moeten daarom op nul worden gezet.

Toongenerator: registers 0,1,2,3,4,5

Om de frequentie in te stellen van de toongenerator schrijft u in de registers 0,1,2,3,4,5. Ieder kanaal kent twee registers om de frequentie te bepalen van een te genereren toon.

Register 0 en 1 voor kanaal A
Register 2 en 3 voor kanaal B
Register 4 en 5 voor kanaal C

Register	Waarde	Beschrijving
0	0-255	Precieze toonfrequentie op kanaal A.
1	0-15	Grove toon op kanaal A.
2	0-255	Precieze toonfrequentie op kanaal B.
3	0-15	Grove toon op kanaal B.
4	0-255	Precieze toonfrequentie op kanaal C.
5	0-15	Grove toon op kanaal C.
6	0-31	Frequentie van de ruisgenerator.

De frequentiestap (f) wordt als volgt berekend:

$$f = (\text{register } 0) + (\text{register } 1) \times 255$$

Deze formule geeft niet de frequentie in Hz, maar hoe groter het frequentiegetal, hoe lager de toon die wordt gegenereerd.
Daarom is de hoogste frequentie die kan worden gegenereerd die in SOUND 0,0: SOUND 0,1 geeft f=0; en de laagste frequentie wordt gegeven door SOUND 0,255:SOUND 1,15.
Om de frequentie in Hz te berekenen moet u de volgende formule gebruiken:

$$f = \frac{178900}{(16 \times Hz)}$$

178900 is de klokfrequentie van PSG (die voor MSX in Engeland kan afwijken).

Wilt u heel nauwkeurig zijn in de toon die u de PSG wilt laten genereren, dan moet u bovenstaande formule hanteren. Nee, niet met een rekenmachine! Gebruik hier uw computer voor.

```
10 INPUT "FREQUENTIE IN HERTZ";HZ
20 F=INT(178900#/(16*HZ))
```

```
30 PRINT "PRECIEZE TOON REGISTER 0, 2 OF 4";F MOD 256
40 PRINT "GROVE TOON REGISTER 1, 3 OF 5";INT(F/256)
```

Bovenstaand programma moet deze truc uitvoeren.
Voorbeeld:
Genereer 2000 Hz geluidsgolf vanuit kanaal A,
f=178900/(16*2000)=55.9

```
10 SOUND 0,55
20 SOUND 1,0
30 SOUND 7,62    mixer
40 SOUND 8,10    geluidssterkte
```

Ruisgenerator-frequentie: register 6

Om de frequentie te veranderen van de witte ruisgenerator, dient u te schrijven naar register 6.
De waarde varieert tussen 0 en 31: hoe lager de waarde, hoe hoger de frequentie. Voorbeeld:

```
10 SOUND 7,55              kanaal A ruis aan
20 SOUND 8,15              geluidssterkte kanaal A op 15
30 FOR F=0 TO 31
40 SOUND 6,F
50 FOR I=1 TO 200:NEXT     vertraging
60 NEXT F
```

U zou nu een langzame afname moeten waarnemen van de witte ruis-frequentie.

Geluidssterkte: registers 8, 9 en 10

De geluidssterkte van het geluid dat wordt uitgezonden vanuit de kanalen A, B en C wordt achtereenvolgens bestuurd door de registers 8, 9 en 10.

```
geen geluid       maximaal volume
0 <----------------------> 15
register 8   =    kanaal A
register 9   =    kanaal B
register 10  =    kanaal C
```

Als in deze registers 16 wordt geschreven, dan wordt de envelope gedeactiveerd. Dit betekent dat de geluidssterkte varieert volgens de vorm van de envelope en de frequentie. De envelope wordt bepaald door de registers 11, 12 en 13.
U moet goed onthouden dat het volume van de TV-luidspreker afhangt van de volumeregeling op de TV. Is de volumeregeling uitgeschakeld dan hoort u

niets. Het geluidsvolume van de MSX-uitvoer hangt ook af van de externe versterker.

Hoe envelopes worden gebruikt: registers 11, 12 en 13

Als envelope niet wordt gebruikt blijft de geluidssterkte constant op het niveau dat wordt aangegeven door de registers 8, 9 en 10. Dit is nogal vervelend, omdat alles wat u hoort een constante toonhoogte is. Echter, de AY-3-8910 PSG kent een envelope-mogelijkheid die het volume periodiek kan laten variëren volgens de beschikbare envelope-vormen.
Envelope wordt ge-activeerd door het op 16 zetten van de volumeregisters 8, 9 of 10. Indien de envelope wordt aangeschakeld, wijzigt het geluidsvolume volgens de registers 11, 12 en 13.

	ENVELOPE UIT	ENVELOPE AAN
KANAAL A REGISTER 8	0-15	16
KANAAL B REGISTER 9	0-15	16
KANAAL C REGISTER 10	0-15	16

De vorm van het envelope-patroon kan worden geselecteerd door het zetten van register 13 op een getal tussen 0 en 15. Echter, de PSG dupliceert enkele van de patronen, zodat u slechts acht verschillende envelopes kunt krijgen.
Het niveau van wijzigen van het volume of envelope-periode kan worden veranderd met de registers 11 en 12; register 12 geeft een grove periodewijziging en register 11 een precieze periodewijziging.

	waarde
register 11	0-255
register 12	0-255

Periode = (register 12) + 256*(register 11)

Let er op dat u slechts één envelope per keer kunt hebben, maar alle drie de kanalen mogen dezelfde envelope gebruiken.
Zie SOUND in de BASIC-referentiesectie voor de envelope-vormen.

Hoofdstuk 19

Hoe bestanden worden gebruikt

In MSX-BASIC is het mogelijk om gegevens op cassette te bewaren, zoals de inhoud van een string-array, en andersom om gegevens in te lezen van cassette. Deze set van gegevens noemt men een bestand (Eng. file). Er is een reeks van BASIC-opdrachten en functies die u in staat stellen om gegevens weg te schrijven en weer in te lezen van een willekeurige periferie (randapparaat zoals een cassetterecorder of floppy drive).

MAXFILES

Voordat u daadwerkelijk een bestand gaat gebruiken doet u er goed aan om MAXFILES te verhogen; dit is een functie die het maximum aantal bestanden aangeeft dat is toegestaan. De vervangingswaarde van MAXFILES is 1, maar dit is gewoonlijk niet voldoende, omdat u alleen gebruik kunt maken van de cassette als opslagmedium indien MAXFILES 1 is. Het maximum van MAXFILES is 15.
Technisch gesproken heeft MAXFILES tot resultaat dat de grootte van het File Control Block in het geheugen wordt verhoogd. Het File Control Block wordt gebruikt als werkgebied door de MSX-ROM als u bestanden manipuleert. (Zie hiervoor de geheugenorganisatie in hoofdstuk 20.)

OPEN

Om de computer te laten weten dat u wilt beginnen met het lezen en schrijven van bestanden, moet u de OPEN-opdracht gebruiken om aan te geven welk medium hiervoor wordt gebruikt. Dit reserveert een buffer in het File Control Block en bepaalt de I/O-mode (invoer/uitvoer-mode) voor deze buffer.
U moet OPEN hebben uitgevoerd voordat u één van de volgende opdrachten gebruikt:

PRINT #	PRINT # USING
INPUT #	LINE INPUT #

INPUT$# EOF
OPEN "<naam medium>[<naam van bestand>]"[FOR<mode>] AS [#] <bestandsnummer>

Er zijn vier randapparaten gedefinieerd in de huidige versie van MSX, te weten:

CAS:	cassetterecorder
CRT:	tekstscherm
GRP:	grafisch scherm
LPT:	afdrukeenheid

Van de bovenstaande randapparaten hebben alleen CAS: en GRP: een duidelijke toepassing. CAS: gebruikt u om gegevens te lezen of te schrijven van en naar een cassetterecorder. GRP: wordt gebruikt om gegevens op een grafisch beeldscherm af te drukken. Het is niet mogelijk om PRINT te gebruiken in schermmode 2 en 3, daarom moet u dus gegevens in de vorm van een bestand naar het scherm schrijven. Een nauwkeurige beschrijving over hoe u op een grafisch scherm kunt afdrukken vindt u in hoofdstuk 14.
Er zijn drie I/O-modes (invoer/uitvoer-modes):

OUTPUT	Specificeert sequentiële uitvoer-mode, dit is het schrijven naar een medium met PRINT# en PRINT # USING.
INPUT	Specificeert sequentiële invoer-mode, dit is lezen van een medium met INPUT #, enz.
APPEND	Specificeert sequentiële toevoegingsmode.

<bestandnummer> is een integere waarde waarvan de waarde ligt tussen één en MAXFILES, het maximum toegestane aantal bestanden. Bestandnummers worden zodanig opgegeven, dat als ze tussen aanhalingstekens staan in PRINT #-opdrachten, de computer weet naar welk medium moet worden geschreven of waar vandaan moet worden gelezen. Het <bestandnummer> wordt geassocieerd met een bestand, zolang als dit is geopend. Het moet er niet één zijn die al eerder is gebruikt in het programma.

Voorbeeld:
Gesteld, we willen een bestand op cassette openen om er vervolgens informatie naar toe te schrijven. De OPEN-opdracht wordt dan als volgt:

```
OPEN "CAS:INFO"FOR OUTPUT AS #1
```

CLOSE

De tegenovergestelde functie van OPEN is de CLOSE-opdracht. U doet er goed aan een bestand te sluiten met CLOSE zodra u er mee klaar bent.

PRINT #

Om naar verschillende randapparaten te schrijven, gebruiken we een speciale PRINT-opdracht, namelijk PRINT #. Het '#'-symbool wordt gevolgd door een bestandnummer dat stond gespecificeerd in de OPEN-opdracht. Gebruikt u PRINT # met een cassette als medium, dan heeft dit het effect van een opname van de gegeven informatie.

PRINT # USING

Dit is de bestandsversie van PRINT USING en heeft dus hetzelfde effect. Hoofddoel is het afdrukken op het grafische scherm in een gespecificeerde vorm.

INPUT #

Om gegevens in te lezen van randapparatuur, anders dan van het toetsenbord, gebruiken we de INPUT #-opdracht. Het effect is identiek aan de INPUT-opdracht, maar in plaats van het intikken van gegevens via het toetsenbord, wordt nu de informatie gelezen van het medium zoals stond gespecificeerd in de OPEN-opdracht; in de meeste gevallen zal dit een cassette zijn.
Gedurende de invoer van numerieke variabelen worden leidende spaties, 'wagen terug'-tekens en linefeed-tekens genegeerd. Hetzelfde geldt voor string-variabelen. Aanhalingstekens worden ook genegeerd.

INPUT$(#)
LINE INPUT#

Dit zijn de bestandsversies van respectievelijk INPUT$ en LINE INPUT (zie ook INPUT #).

EOF

EOF geeft aan of het einde van een bestand is bereikt. Is het einde bereikt dan geeft EOF -1 terug, zoniet dan is het resultaat altijd 0.

Voorbeeldprogramma:
Hier volgt een demonstratieprogramma dat ruwweg een idee geeft hoe bestanden kunnen worden gebruikt.
Het programma vraagt u vijf namen in te voeren en slaat deze vervolgens op in een cassettebestand. Daarna wordt u gevraagd om de gegevens terug te laden.

```
10 DIM N$(5)
20 MAXFILES=4
30 FOR I=1 TO 5
40 INPUT "UW NAAM SVP";N$
50 NEXT I
60 PRINT "STAAT CASSETTERECORDER AAN? (J=OK)"
70 IF INKEY$<>"j" THEN 70
80 OPEN "CAS:" FOR OUTPUT AS #2
90 FOR I=1 TO 5
100 PRINT #2,N$(I)
110 NEXT I
120 CLOSE#2
130 PRINT "KLAAR VOOR LADEN? (J=OK)"
140 IF INKEY$<>"j" THE 140
150 OPEN "CAS:" FOR INPUT AS #3
160 FOR I= 1 TO 5
170 INPUT #3,N$(I)
180 NEXT I
190 CLOSE #3
200 FOR I=1 TO 5
210 PRINT N$(I)
220 NEXT I
```

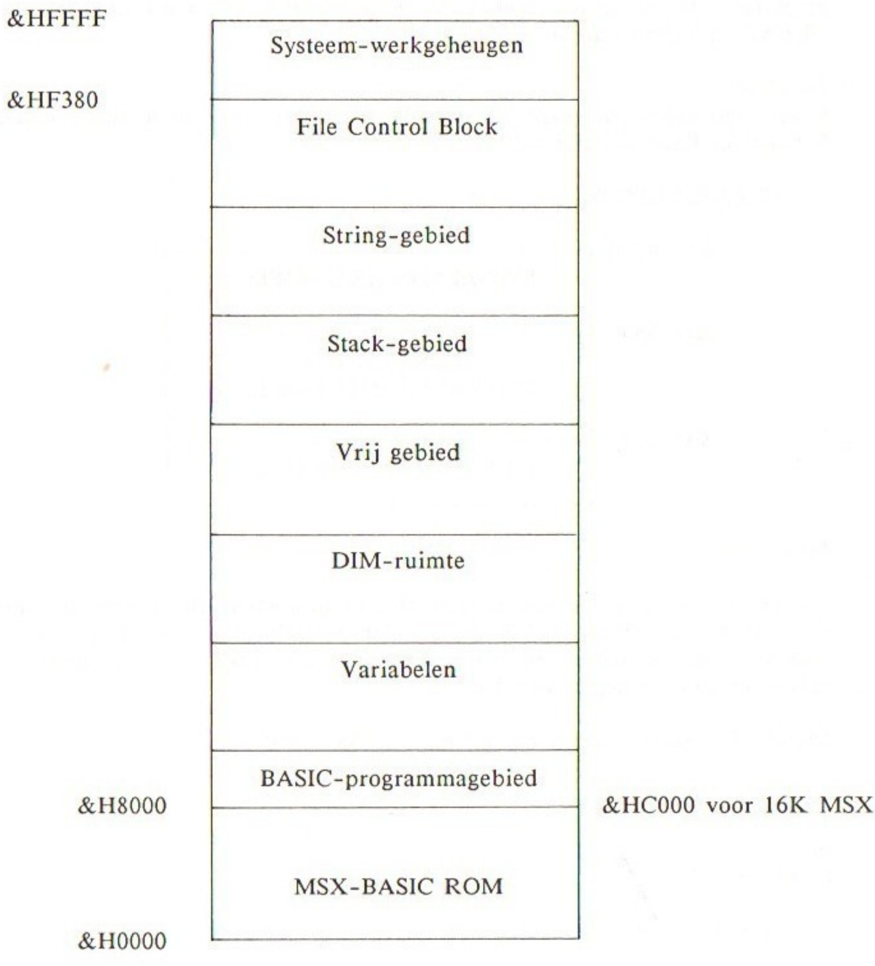
REGEL 10	Array voor namen.
REGEL 20	Verhoog maximum aantal bestanden.
REGEL 30	Voer vijf namen in zodat we wat kunnen doen.
REGEL 60	Activeer cassette.
REGEL 70	Wacht op cassette-AAN.
REGEL 80	Open een bestand naar cassette.
REGEL 90	Uitvoer naar cassette met PRINT #.
REGEL 120	Sluit (CLOSE) bestand na gebruik.
REGEL 130	Spoel cassette terug opdat we de namen kunnen laden.
REGEL 150	Opent een bestand voor invoer.
REGEL 160	Leest een cassettebestand met INPUT #.
REGEL 190	Sluit bestand na gebruik.
REGEL 200	Drukt het resultaat af op het scherm.

```
RUN
UW NAAM SVP? TOM
UW NAAM SVP? PAUL
UW NAAM SVP? BELLA
UW NAAM SVP? JANE
UW NAAM SVP? CHRIS
STAAT CASSETTERECORDER AAN? (J=OK) druk op RECORD
```

```
KLAAR VOOR LADEN? (J=OK)  REWIND en druk dan op PLAY
TOM
PAUL
BELLA
JANE
CHRIS
OK
```

Hoofdstuk 20

Geheugenorganisatie



Werkgebied

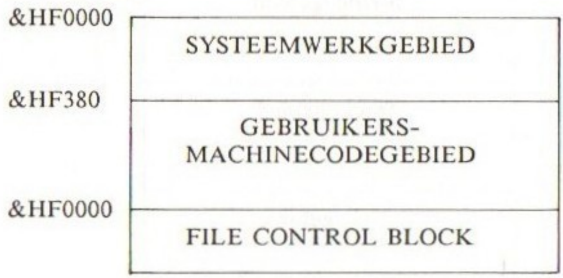
Het werkgebied bevindt zich tussen de geheugenlocaties &HFFFF en &HF380 en wordt gebruikt door de MSX-BASIC ROM voor interne bewerkingen. Dit gebied bevat de verschillende systeemvariabelen, waarvan u een overzicht vindt in het gedeelte over de BIOS.

File Control Block

Dit gebied is gereserveerd voor invoer/uitvoer-bewerkingen met bestanden. Opdrachten zoals PRINT # en INPUT # benutten dit gebied.
De grootte van dit blok kan worden gewijzigd met MAXFILES. De uiterste waarde van dit blok is &HF380. U kunt deze waarde echter verlagen om machinecode- of gegevensgeheugen te reserveren. Dit kunt u bereiken met CLEAR, die deze waarde verlaagt.

Voorbeeld:
Creëer een geheugengebied om gebruikers-machinecode op te slaan, tussen de locaties &HF380 en &HF0000:

```
CLEAR,&HF0000
```



String-gebied

Dit gebied verzorgt de opslag van string-variabelen. De vervangingswaarde van dit string-gebied is 200 bytes. Dit bepaalt tevens de limiet van het maximum aantal tekens in een string (dus 200 tekens). Deze grootte kan echter worden aangepast met CLEAR.

Voorbeeld: verhoog het string-gebied tot 255 bytes:

```
CLEAR 255
```

Stack-gebied

Dit gebied verzorgt de opslag voor stacks die nodig zijn voor FOR...NEXT-

lussen en GOSUB's. Dit is de opslag van adressen waarop de BASIC kan terugkeren na de uitvoering van NEXT of RETURN.

Vrij gebied

Dit is een ongebruikt gebied. De grootte kan worden gevonden met behulp van de FRE-functie, zoals hieronder blijkt:

PRINT FRE(0)

28815	voor 32K of 64K MSX-systemen na het aanzetten van de machine.
12431	voor 16K MSX-systemen na het aanzetten van de machine.

Let er op dat deze waarde wordt verlaagd als u een BASIC-programma en variabelen inbrengt.

DIM-ruimte

Dit gebied verzorgt de opslag van DIM-arrays en wordt verhoogd na uitvoering van een DIM-opdracht. Als een array een string-array blijkt te zijn, worden verwijzingen naar de string opgeslagen.

Variabelengebied

Dit gebied verzorgt de opslag van numerieke variabelen en een string-verwijzing naar het string-gebied. Het adres van de gegevens van een specifieke variabele kan worden gevonden met de VARPTR-functie. (Zie hiervoor VARPTR in het BASIC-verwijzingsgedeelte.)

BASIC-programmagebied

Hierin wordt uw BASIC-programma opgeslagen.

Hoofdstuk 21

USR-functie en machinecode

Om machinecode-subroutines uit te voeren vanuit BASIC, wordt de USR-functie gebruikt. De USR roept een machinecode-programma aan op een locatie zoals opgegeven in de DEF USR-opdracht. U kunt 10 machinecode-programma's tegelijkertijd gebruiken. Het is mogelijk om meer dan 10 routines te gebruiken door het opnieuw definiëren van de USR-functie.

Hoe wordt USR gedefinieerd?

Om een machinecoderoutine aan te roepen moet de computer weten wat het beginadres van deze routine is. Gebruik DEF USR om een adres te koppelen aan USR.
Voorbeeld:
Gesteld, de machinecode begint vanaf positie &HF000:

```
DEF USR0=&HF000
```
(u mag 0 afkorten, dus DEF USR=&HF000)

Hoe een machinecoderoutine wordt uitgevoerd

De USR kan op dezelfde manier worden aangewend als de gewone BASIC-functies. Dit betekent dat ze kan worden gebruikt in een uitdrukking, zoals hieronder blijkt:

```
X=USR(99)
Y$="M/K"+USR9("ABX")
PRINT USR7(0)
```

Het bovenstaande voert de subroutine uit en geeft een waarde terug afhankelijk van wat de machinecode doet. De numerieken of strings tussen haakjes zijn parameters die doorgegeven kunnen worden aan de machinecode. Aldus is de wijze waarop de USR-functie wordt gebruikt afhankelijk van uw machinecoderoutine. U kunt een argument meegeven, maar ook een dummy-parameter (nep-parameter); ook kan een parameter worden teruggegeven of helemaal niets.

Wilt u machinecode zonder extra parameters of uitvoer laten uitvoeren, gebruik dan een dummy, zoals hieronder staat:

```
DUMMY=USR0(0)
```

DUMMY is een nep-variabele en (0) is een nep-parameter.

Hoe wordt een parameter overgedragen aan een machinecoderoutine vanuit BASIC?

U kunt iedere mogelijke soort parameter vanuit BASIC overdragen aan de machinecode met behulp van een USR<nummer>(<parameter>). De parameter moet tussen haakjes staan. Als MSX de functie uitvoert dan controleert hij het type argument, dus of deze integer, enkelvoudig nauwkeurig of dubbel nauwkeurig dan wel een string is.
Blijkt er een parameter te zijn meegegeven, dan moet de machinecode weten van welke soort deze is en waar ze zich bevindt in het geheugen. MSX-BASIC bepaalt eerst de parameter voordat de machinecode wordt uitgevoerd.
Om uit te vinden welke soort parameter is overgedragen, wordt gekeken op positie &HF663:

&HF663 = 2 = 00000010 integere parameter
&HF663 = 4 = 00000100 enkelvoudige nauwkeurige parameter
&HF663 = 8 = 00001000 dubbel nauwkeurige parameter
&HF663 = 3 = 00000011 string

Locatie van meegegeven parameter:

integer (&HF663=2)
&HF7F8=<lagere byte>
&HF7F9=<hogere byte>
<integere parameter>=<lagere byte>+256*<hogere byte>

Enkelvoudig nauwkeurig getal (&HF663=4).

&HF7F6 =]
&HF7F7 =] gegevens opgeslagen
&HF7F8 =] in BCD vanaf &HF7F6
&HF7F9 =]

Dubbel nauwkeurig getal (&HF663=8).

&HF7F6 =]
&HF7F7 =]
&HF7F8 =]
&HF7F9 =] gegevens opgeslagen
&HF7FA =] in BCD vanaf &HF7F6
&HF7FB =]

```
&HF7FC = ]
&HF7FD = ]
```

String (&HF663=3).

```
&HF7F8  = ] laag  ]
&HF7F9  = ] hoog  ]   ...<adres1> adres van string-beschrijving

<adres1>    = lengte van string
<adres1>+1  = laag  ]
<adres1>+2  = hoog  ]   ...<adres2>, string-positie
```

Hoe wordt een parameter teruggegeven vanuit een machinecoderoutine

Om een parameter aan BASIC terug te geven vanuit een machinecoderoutine, dient u &HF663 op de juiste waarde te zetten voor het soort gegeven en moet u de parameters zelf opslaan in de relevante adressen, voordat u de machinecode verlaat. BASIC pakt alles op wat zich in de parameterbuffer bevindt en verandert dit in een variabele na het verlaten van de machinecode.

Hoofdstuk 22

MSX-geheugenbesturing en cartridge-slotmechanisme

Inleiding

MSX's CVE, de Z80 microprocessor, heeft een 16-bit adresbus die 64Kbytes geheugen kan adresseren op ieder moment. 64Kbytes geheugen is verdeeld in vier 16K pagina's geheugen. De MSX kan het geheugen uitbreiden door over te schakelen naar verschillende 'slots' voor iedere pagina. Een 'slot' is in principe een ruimte van 64K en kan worden voorgesteld als een bank van gescheiden geheugen.
Fysiek kan dit slot een insteekcartridge zijn, of een hardwarebank met geheugen, zoals de MSX-BASIC ROM en RAM.
Om te selecteren welk slot wordt gebruikt voor welke pagina, dient het slot-selectieregister te worden gezet. Hierover later meer.
Gewoonlijk heeft MSX de volgende geheugenorganisatie:

PAGINA	SLOT 0 SYSTEEMSLOT	SLOT 1, 2 OF 3 CARTRIDGESLOT DAT KAN WORDEN INGESTOKEN
PAGINA 3	16K RAM voor 16K MSX-computer	
PAGINA 2	16K RAM voor 32K MSX-computer	ROM-cartridge voor spellen software 8 of 16K ROM, of 16K uitbreiding RAM voor 16K MSX-systeem
PAGINA 1	MSX-BASIC ROM	gebruikt voor BASIC-uitbreiding ROM of disk-besturings-ROM of andere talen
PAGINA 0	MSX-BASIC ROM (BIOS)	

Cartridge

Alle MSX-computers hebben ten minste één MSX-standaard-cartridgeslot waar u verschillende apparaten op kunt aansluiten. Hier volgt een lijst van verschillende randapparaten:

1. Uitbreidings-RAM. Gewoonlijk om 16K machines uit te breiden naar 32K.
2. Spelprogramma-uitbreidings-ROM. De cartridge bevat óf een machinecode-programma of een BASIC-programma.
3. BASIC-uitbreidings-ROM. Deze ROM bevat routines waarmee de MSX-BASIC wordt uitgebreid. De ROM-routines worden aangeroepen met CALL-opdrachten. Sommige cartridges beschikken over hun eigen RAM als werkgeheugen.
4. Invoer/uitvoer-cartridge. Dit kan een floppy disk-besturing zijn of een printer-interface of een lichtpen-cartridge. Meestal wordt de eigen hulproutines-ROM er bij geleverd om de in-/uitvoer (I/O) te besturen.

Ieder van deze bovenstaande mogelijkheden kan in elk van de slots worden gestoken. MSX beschikt over een slot-selectiemechanisme, zodat de computer weet welke cartridge moet worden benaderd.

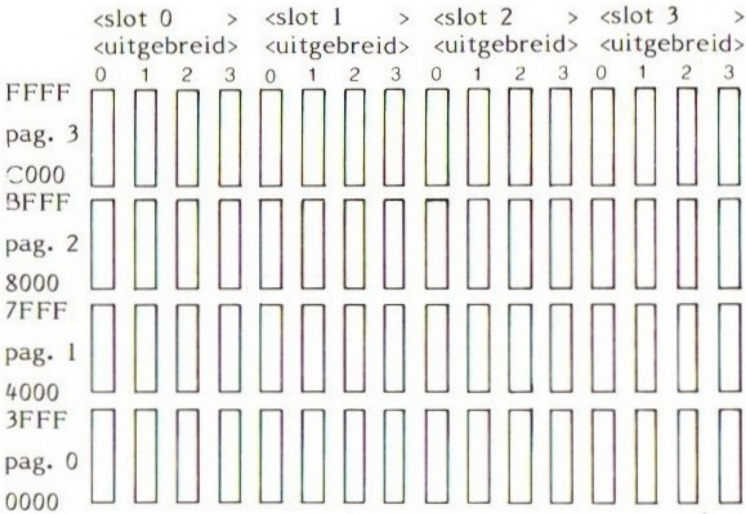
Basisslot-arrangementen

MSX kan vier basisslots hebben, waarvan slotnummer 0 de MSX-BASIC ROM is; deze wordt daarom systeemslot genoemd. Elk van deze vier slots kan worden uitgebreid naar vier uitbreidingsslots. Daardoor is het totale aantal slots uit te breiden tot 16. Deze 16 slots kunnen ieder 64K RAM zijn, waardoor u 1 Mbyte geheugen wordt aangeboden. Dit is tevens de maximum RAM welke MSX aankan. U dient er echter rekening mee te houden dat u alle bytes niet in één keer vanuit BASIC kunt bereiken! U heeft hiervoor óf MSX-DOS óf een machinecoderoutine nodig om effectief gebruik te kunnen maken van geheugen-ruimte groter dan 64K.

Basisslot-configuratie:

	SLOT0	SLOT1	SLOT2	SLOT3	
FFFF C000	PAG. 3	PAG. 3	PAG. 3	PAG. 3	16K
BFFF 8000	PAG. 2	PAG. 2	PAG. 2	PAG. 2	16K
7FFF 4000	PAG. 1	PAG. 1	PAG. 1	PAG. 1	16K
3FFF 0000	PAG. 0	PAG. 0	PAG. 0	PAG. 0	16K

Uitgebreide slotconfiguratie:



Slotselector

MSX kan verschillende slots of geheugenbanken hebben, maar tenzij ze worden bestuurd door andere middelen, kunnen ze elkaar tegenwerken. Om te administreren welk slot voor welke pagina wordt gebruikt, heeft MSX een mechanisme om slots te selecteren. Op ieder moment kan de CVE 64K of vier pagina's RAM benaderen. Deze vier pagina's kunnen ieder verschillende slots zijn. Deze pagina's kunnen zich in verschillende slots bevinden. De pagina's in de slots worden geselecteerd met de 8255 PPI's uitvoerpoort A (zie hiervoor het slotselectie-circuit).

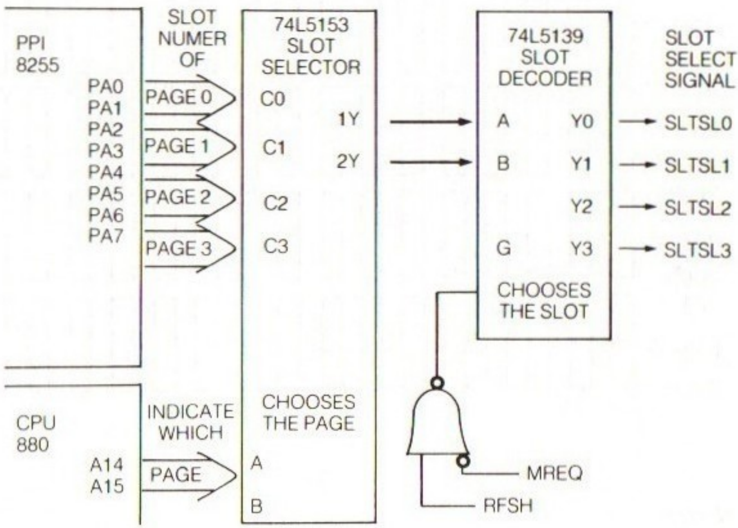
PA0 en PA1 geven het slotnummer voor pagina 0
PA2 en PA3 geven het slotnummer voor pagina 1
PA4 en PA5 geven het slotnummer voor pagina 2
PA6 en PA7 geven het slotnummer voor pagina 3

Het signaal van de PPI wordt gezonden naar de slotselector 74LS153. Deze chip ontvangt ook signalen van de CVE van de Z80 ten behoeve van de huidige pagina welke de CVE leest of schrijft.

0	0	pagina 0
0	1	pagina 1
1	0	pagina 2
1	1	pagina 3

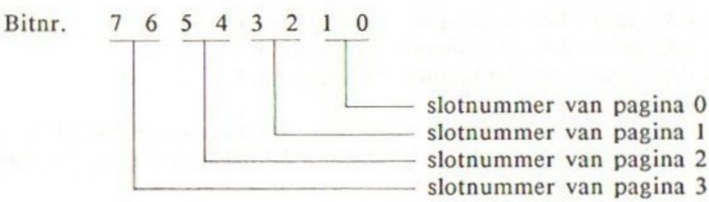
Het geselecteerde slotnummer wordt dan overgedragen aan de 2 naar 4-decoder, 74LS139, welke het slotselectiesignaal geeft, SLTSL, aan het slot dat hoort bij de vier pagina's.

SLOTSELECTORCIRCUIT-DIAGRAM



Hoe een slot wordt geselecteerd en ge-activeerd

Om slots te selecteren voor iedere pagina vanuit de software, dient u uitvoer te sturen naar in-/uitvoeradres &HA8, welke de uitvoer is van de PPI naar poort A.
De waarde die wordt uitgevoerd is een 8-bit getal.



Voorbeeld:
Gesteld, u wilt slot 0 gebruiken voor pagina 0, 1 en 3 en slot 1 voor pagina 2.

Bitnr.	7	6	5	4	3	2	1	0		
	0	0	0	1	0	0	0	0	=	16

U kunt 16 uitvoeren naar I/O-adres &HA8. Een betere methode is echter om de BIOS te gebruiken met een CALL ENASLT (&H0024) vanuit machinecode.

Uitbreiding van slots

MSX heeft de beschikking over vier basisslots, van 0 tot 3. Niet ieder daarvan is in elke machine aanwezig, maar ieder basisslot kan worden uitgebreid met een uitbreidingsdoos. Het maximum aantal slots is daarom 16. Omdat ieder daarvan kan worden gevuld met 64K RAM, is de totale capaciteit waarmee MSX overweg kan 1 Mbyte.

De uitbreidingsslots kunnen niet verder worden uitgebreid. De uitbreiding bevindt zich altijd achter een basisslot, dus het heeft geen zin een uitbreidingsdoos in een reeds uitgebreid slot te steken.

Om een uitbreidingsslot te selecteren dient u eerst het basisslot te selecteren waaraan de uitbreiding is gekoppeld door gebruik te maken van poort A van 8255 PPI. Het slotselectie-register bevindt zich op locatie &HFFF in het uitbreidingsslot. Dit bepaalt of de geselecteerde pagina in dat slot wordt gebruikt of niet.

Om uit te vinden of een basisslot een uitbreidingsdoos heeft of niet, moet u schrijven naar locatie &HFFFF (POKE &HFFFF,&H0F). Als u dan vervolgens &HFFFF leest (PRINT PEEK(&HFFFF)) en daarop de complementswaarde krijgt van wat u had weggeschreven, betekent dit dat er een uitbreiding aanwezig is.

Buffer van uitbreidingsslot

De basis-slots hebben geen buffer nodig, maar de uitbreidingen ervan wel. Daarom heeft een cartridgeslot-uitbreidingsdoos een tweeweg gegevensbus-buffer binnen in zijn circuit.

De buffer verandert van richting, afhankelijk van óf er geschreven óf er gelezen wordt. Het besturingssignaal dat de richting van de buffer bestuurt is BUSDIR. Die randapparaten die signalen sturen naar de CVE, zoals een I/O-cartridge, dienen eerst een BUSDIR-signaal te versturen om de richting van de buffer aan te passen van de uitbreidingsslots naar de CVE. Die cartridges die alleen RAM of ROM bevatten hoeven het BUSDIR-signaal niet te manipuleren.

RAM-zoekprocedure

Zodra de MSX-computer wordt aangeschakeld, doorzoekt hij alle slots om de systeem-RAM te selecteren.

1. Allereerst wordt de beschikbare RAM in pagina 2 doorzocht, van &HF8000 tot &HBFFF, en wordt vervolgens het slot met de grootste RAM in pagina

2 geactiveerd. Zijn er meerdere slots met de grootste RAM-capaciteit, dan wordt het slot met het kleinste slotnummer gekozen.
2. Hetzelfde wordt daarna voor pagina 3 gedaan, van &HC000 tot &HFFFF. Opnieuw wordt de grootste RAM met het laagste slotnummer geselecteerd.
3. Ten slotte wordt gecontroleerd of het een continue RAM betreft van positie &H8000 tot &HFFFF en de systeemvariabele BOTTOM (&HFC48) op het adres van de laagst beschikbare RAM gezet.

16K UITBREIDING NAAR 32K MACHINE

	SYSTEEMSLOT	CARTRIDGE	
FFFF C000	RAM 16K		PAG. 3
BFFF 8000		RAM 16K	PAG. 2 BOTTOM=&H8000
7FFF 4000	MSX-ROM		PAG. 1
3FFF 0000	MSX-ROM		PAG. 0

Cartridge ROM-software

ROM-zoekprocedure

Nadat de systeem-RAM is geselecteerd, zoekt de MSX naar ROM-cartridges van &H4000 tot &HBFFF, pagina 1 en pagina 2. Hij zoekt naar een correct ID aan het begin van iedere pagina van slot 0 tot slot 3 en de uitbreidingsslots indien aanwezig. Het ID-gebied bevindt zich aan het begin van iedere pagina en kan één van de volgende formaten hebben:

Bovenkant van pagina +&H00	ID "AB"	ID...2 byte-code "AB" om aan te geven dat zich hier een ROM-cartridge bevindt.
+&H02	INIT	INIT bevat het adres van de initialisatieroutine voor die cartridge.

+&H04	STATEMENT	STATEMENT bevat het adres van de routine die de uitbreiding van opdrachten behandelt.
+&H06	DEVICE	DEVICE bevat het adres van de routine voor uitbreiding van randapparaten.
+&H08	TEXT	TEXT bevat het startadres van het BASIC-programma in de cartridge.
+&H0A	gereserveerd	
+&H10		

Opmerking: ID, INIT, STATEMENT, DEVICE en TEXT bevatten nullen indien niet van toepassing.

MSX-BASIC neemt de volgende actie bij een cartridge-zoekprocedure:

1. Controleert het ID-gebied en vindt uit welke soort routine deze bevat. Daarna wordt de informatie overgedragen naar het relevante werkgebied.
2. Voert de INIT-routine uit, indien aanwezig.
3. Voert BASIC-programma uit, indien aanwezig.

Opmerking: STATEMENT en DEVICE worden nu nog niet uitgevoerd, omdat deze alleen noodzakelijk zijn zodra de gebruiker de relevante uitbreiding benut.

INIT: initialisatieroutine

INIT bevat het beginadres van de initialisatieroutine voor een specifieke cartridge. Gewoonlijk worden daarbij in-/uitvoerapparaten aan de cartridge gekoppeld, of wordt een hook opgezet (meer over 'hook' later in het gedeelte over Hook), of er wordt een werkruimte gecreëerd voor de cartridge-software.
De initialisatieroutine kan alle registers wijzigen behalve de stackpointer [SP]. Er wordt teruggekeerd naar BASIC na het uitvoeren van de Z80 RET-instructie.
De INIT hoeft geen initialisatieroutine te zijn. Het kan een machinecode-programma zijn, dat onmiddellijk na het inschakelen van de computer moet worden uitgevoerd. Dit zou een spel kunnen zijn.
Is er geen initialisatieroutine aanwezig dan staan er nullen in INIT.

TEXT: BASIC-programma

Een ROM-cartridge hoeft niet noodzakelijkerwijs een machinecodeprogramma te zijn: het zou in BASIC kunnen zijn geschreven. TEXT bevat het startadres van het BASIC-programma in de cartridge.
Bij het programmeren van cartridge-software in BASIC, moeten de volgende punten goed worden onthouden:

1. Zijn er meerdere cartridges aanwezig met BASIC-software, dan voert de machine alleen dat programma uit dat zich bevindt in het slot met het laagste slotnummer.
2. Het BASIC-programma in de cartridge staat opgeslagen in de verkorte vorm (enkele symbolen).
3. De cartridge moet zijn geplaatst in pagina 2, &H8000 tot &HBFFF. Dit betekent dat de maximum BASIC-geheugenruimte in de cartridge 16K is.
4. RAM op pagina 2 van een ander slot wordt uitgeschakeld en kan dus niet worden gebruikt vanuit het BASIC-cartridge-programma.
5. Het door TEXT gerefereerde adres moet 0 bevatten.
6. De regelnummers voor GOTO's en GOSUB's moeten worden gewijzigd in pointers voor snellere uitvoering.

Bevindt zich geen BASIC-programma in de cartridge, dan bevat TEXT nullen.

STATEMENT: Routine voor uitbreiding van opdrachten

Door gebruik te maken van de CALL-opdracht in MSX-BASIC, kunt u het aantal BASIC-opdrachten uitbreiden dat staat opgeslagen buiten de MSX-BASIC-ROM. Een cartridge die een uitgebreide BASIC-opdracht bevat moet het startadres van de eerste uitgebreide opdrachtroutine bevatten in STATEMENT. De cartridge moet worden geplaatst in pagina 1, maar kan in feite ieder slot zijn, behalve het systeemslot waar BASIC zich bevindt.
De syntax voor een uitgebreide opdracht is:

```
CALL <naam van opdracht>
CALL <naam van opdracht>(argument)
```

CALL kan worden afgekort met het onderstrepingsteken '_'. Zodra BASIC een CALL onderkent, slaat het de naam van de uitgebreide opdracht op in het systeem-werkgebied, PROCNM. PROCNM bevindt zich op de locatie &HFD89 en verder en is 16 bytes lang. De naam van de opdracht eindigt op een nul indien opgeslagen in PROCNM, zodat de maximumlengte van een uitbreidings-opdracht 15 tekens kan zijn.
Voordat de uitgebreide opdracht wordt uitgevoerd verwijst de tekstpointer, dit is het HL-registerpaar, naar het adres achter de naam van STATEMENT.
De STATEMENT-procedure in de cartridge controleert vervolgens de inhoud van PROCNM en voert de routine uit die past bij de naam van de opdracht. Nadat de uitgebreide opdracht is uitgevoerd wordt de carry flag gewist en de

tekstpointer [HL] op de positie van de volgende opdracht gezet.
Wordt er geen gelijkheid gevonden bij het zoeken naar de naam van de uitgebreide opdracht in PROCNM, dan blijven de carry flag en [HL] ongewijzigd en wordt er teruggekeerd naar BASIC met een 'Syntax Error'-melding. Bevindt zich geen uitbreidingsroutine in de cartridge dan bevat STATEMENT nullen.

DEVICE: routine die uitbreiding van randapparaten behandelt

MSX-BASIC heeft de mogelijkheid om een uitbreiding van een invoer/uitvoer-medium te koppelen aan een cartridge-slot. DEVICE bevat het adres van de routine die de uitbreiding van dergelijke apparaten behandelt.
De volgende opmerkingen moeten goed worden onthouden bij gebruik van de DEVICE-routines:

1. De cartridge moet worden geplaatst op pagina 1, &H40000-&HBFFF.
2. Een cartridge (16K) kan tot vier logische apparaataansluitingen hebben.
3. De naam van het apparaat staat opgeslagen in het systeemgebied PROCNM, zoals dat het geval was bij STATEMENT. PROCNM bevindt zich vanaf &HFD89 en verder en is 16 bytes lang. De naam van het apparaat eindigt op een nul als deze is opgeslagen in PROCNM, zodat de maximum naamlengte 15 tekens is.
4. Indien BASIC een naam van een randapparaat ontdekt in een OPEN-opdracht of een andere opdracht die niet bekend is bij de MSX-ROM, wordt de routine van DEVICE aangeroepen met in register A &HFF geladen. Wordt er geen gelijkheid geconstateerd, dan wordt de carry flag gezet. Is het apparaat wel aanwezig, dan wordt ID (van 0 tot 3) teruggegeven in register A en de carry flag teruggezet. Alle registers zijn aan verandering onderhevig gedurende de routine.
5. Een echte invoer/uitvoer-opdracht (I/O) wordt uitgevoerd als register A is geladen met:

0	Open
2	Close (afsluiten)
4	Willekeurig I/O
6	Sequentiële uitvoer
8	Sequentiële invoer
10	LOC-functie
12	LOF-functie
14	EOF-functie
16	FPOS-functie
18	Back-up-teken

De systeemvariabele DEVICE moet worden gezet op het ID-nummer van het randapparaat (0-3).
Is er geen DEVICE-routine aanwezig in de cartridge, dan moet DEVICE nullen bevatten.

Beschrijving van de systeemvariabelen betreffende het slotmechanisme

Status van ieder slot

EXPTBL - geeft aan welk slot is uitgebreid.
Locatie - &HFCC1 - 4 bytes lang.

EXPTBL &HFCC1 voor slot 0
&HFCC2 voor slot 1
&HFCC3 voor slot 2
&HFCC4 voor slot 3
&H80 geeft uitgebreid slot aan
&H00 niet uitgebreid

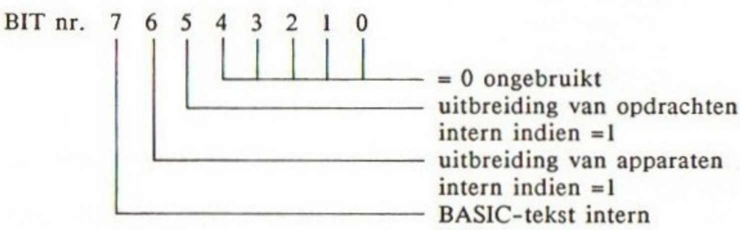
SLTTBL - geeft aan welke waarde op dit moment uitvoer is naar het uitbreidingsslot-selectieregister. Dit is alleen geldig indien de bijbehorende EXPTBL &H80 bevat, dit is als een slot inderdaad is uitgebreid.
Locatie - &HFFC5 - 4 bytes lang.

SLTTBL &HFFC5 voor slot 0
&HFCC6 voor slot 1
&HFCC7 voor slot 2
&HFCC8 voor slot 3

Status van iedere pagina

SLTATR - bevat wat zich in iedere pagina bevindt voor ieder van de mogelijke pagina's.
Locatie - &HFCC9 - 64 bytes lang.

SLTATR &HFCC9 voor basisslot 0 uitbreidingsslot 0 pagina 0
&HFCCA voor basisslot 0 uitbreidingsslot 0 pagina 1
...
...
&HFD07 voor basisslot 3 uitbreidingsslot 3 pagina 2
&HFD08 voor basisslot 3 uitbreidingsslot 3 pagina 3



SLTWRK - werkgebied voor iedere pagina. Het gebruik hangt af van wat er in

de pagina staat, maar 2 bytes per pagina zijn toegestaan.
Locatie - &HFD09 - 128 bytes lang; 2 bytes per pagina.

SLTWRK	&HFD09 voor	basisslot	0	uitbreidingsslot	0	pagina	0
	&HFD0A voor	basisslot	0	uitbreidingsslot	0	pagina	0
	&HFD0B voor	basisslot	0	uitbreidingsslot	0	pagina	1
	&HFD0C voor	basisslot	0	uitbreidingsslot	0	pagina	1
	...						
	...						
	&HFD85 voor	basisslot	3	uitbreidingsslot	3	pagina	2
	&HFD86 voor	basisslot	3	uitbreidingsslot	3	pagina	2
	&HFD87 voor	basisslot	3	uitbreidingsslot	3	pagina	3
	&HFD88 voor	basisslot	3	uitbreidingsslot	3	pagina	3

Hoofdstuk 23

Randapparatuur

Cassetterecorder

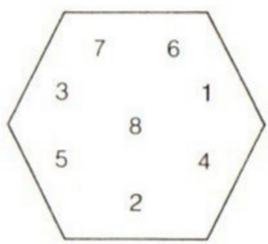
De volgende opdrachten en functies behoren bij het gebruik van een cassetterecorder (gedetailleerde beschrijvingen staan in het BASIC-verwijzingsgedeelte).

CLOAD
CSAVE
MOTOR
SAVE
LOAD
BSAVE
BLOAD

Connector: 8 pins DIN-plug, type 45326

Signaaltabel:

NR	SIGNAAL	RICHTING
1	GND	...
2	GND	...
3	GND	...
4	CMTOUT	Uitvoer
5	CMTIN	Invoer
6	REM+	Uitvoer
7	REM-	Uitvoer
8	GND	...



Printer

Indien uw MSX-computer is voorzien van een printer-interface, kunt u ieder type printer gebruiken die is voorzien van een Centronics-interface. Heeft uw computer geen ingebouwde printer-interface, dan kunt u een printer-interface-cartridge kopen die in een cartridgeslot kan worden geplugd.

Sommige printers zijn voorzien van een set MSX-standaardtekens. Deze bijpassende MSX-printers zijn bij enkele leveranciers beschikbaar en bieden het voordeel dat ook alle grafische MSX-tekens kunnen worden afgedrukt. Een niet-MSX-printer zal niet in staat zijn deze tekens af te drukken; in de plaats daarvan zal er een spatie voor ieder grafisch teken worden afgedrukt.
Gebruikt u een MSX-printer, dan moet u dit de computer laten weten met behulp van de SCREEN-opdracht:

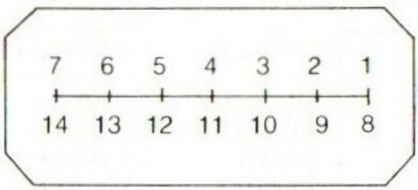
SCREEN,,,,0 selecteert MSX-printer
SCREEN,,,,1 selecteert niet-MSX-printer

LPRINT
LPRINT USING

Om naar een printer te schrijven dient u LPRINT- en LPRINT USING-opdrachten te gebruiken, in plaats van PRINT en PRINT USING. De syntax is volledig identiek, alleen zullen LPRINT en LPRINT USING niets op het scherm afdrukken. Wilt u zowel op het scherm als via de printer afdrukken, gebruik dan zowel PRINT als LPRINT in uw programma.
LLIST biedt een afdruk van de programmalijst. De syntax is identiek aan het LIST-commando.
De LPOS-functie geeft de positie aan van de printerkop, en is identiek aan de POS-functie, die u de positie van de cursor geeft op het scherm.
Meer gegevens over de printer worden meegeleverd als u een printer aanschaft. Vaak worden voorbeeldprogramma's meegeleverd, geschreven in Microsoft BASIC, dat past bij MSX-BASIC.

Connector: Centronics type, 14 pins Aphenol.

PIN NR	SIGNAALNAAM
1	PSTB
2	PDB0
3	PDB1
4	PDB2
5	PDB3
6	PDB4
7	PDB5
8	PDB6
9	PDB7
10	NC
11	BUSY
12	NC
13	NC
14	GND



Joystickpoort

De meeste MSX-computers beschikken over twee joystick-aansluitingen, terwijl weer andere er slechts één hebben; BASIC kan er twee aan. Naast joysticks kunt u andere apparaten inpluggen, zoals 'touch pads' en 'game paddles'.
Connector: AMP 9 pins.

Joystick

Joysticks worden hoofdzakelijk gebruikt voor spellen. Beschikt uw computer over twee joystick-aansluitingen, maar heeft u maar één joystick, wees er dan zeker van dat u deze inplugt in joystick-poort 1, omdat de meeste commerciële programma's veronderstellen, bij het gebruik van een joystick, dat deze is aangesloten op deze poort.
Om de status te lezen van de joystick vanuit BASIC, kunt u STICK- en STRIG-opdrachten gebruiken. STICK geeft de richting aan van de joystick en STRIG geeft u de status van de vuurknop op de joystick.
De ON STRIG GOSUB-opdracht biedt een interrupt-mogelijkheid. Zie hiervoor de Evenementen- en Interruptie-sectie.

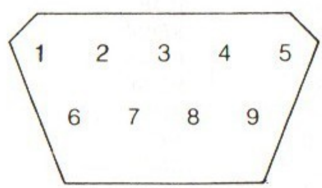
Game paddle

U kunt tot 12 game paddles inpluggen, zes aan ieder van de joystick-poorten. De status van de paddle kan worden gelezen met behulp van de PDL-functie (zie PDL).

Touch pad

U kunt één touch pad per joystickpoort inpluggen.
PAD geeft de status van een touch pad (zie PAD in het BASIC-verwijzings-gedeelte).

	SIGNAAL	RICHTING
1	VOORUIT	INVOER
2	ACHTERUIT	INVOER
3	LINKS	INVOER
4	RECHTS	INVOER
5	+5V	...
6	TRG 1	INVOER
7	TRG 2	INVOER
8	UITVOER	UITVOER
9	GND	...



Deel 3
Verwijzingen

Hoofdstuk 1

BASIC-sleutelwoorden

Verklaring
BASIC-sleutelwoordformaat

Dit hoofdstuk bevat alle MSX-BASIC-sleutelwoorden op alfabetische volgorde, daarbij vindt u alles wat u erover dient te weten. De beschrijvingen van de sleutelwoorden worden in een standaardvorm gepresenteerd, zodat het eenvoudiger is om de bedoeling te vinden. De beschrijving van ieder sleutelwoord wordt als volgt opgedeeld:

SLEUTELWOORD
Dit wordt meestal gevolgd door een aantal woorden die aangeven waarvan het woord is afgeleid.

BESCHRIJVING
Dit gedeelte verklaart in eenvoudige bewoordingen wat het sleutelwoord doet.

SYNTAX
Iedere mogelijke syntax wordt opgesomd, met daarbij de verklaringen van de uitvoeringen en de verschillen tussen de diverse vormen.
De volgende symbolen worden hierbij gebruikt:

<num const>	een gewoon getal, zoals 100 of 1.414.
<num var>	een numerieke variabele, zoals X of BEDRAG.
<num exp>	geeft een uitdrukking aan die resulteert in een getal, zoals 2*PI*R*COS(Y). <num const> en <num var> kunnen worden beschouwd als eenvoudige uitdrukkingen, en kunnen daarom altijd worden toegepast als een uitdrukking mogelijk is.
<numeriek>	kan ieder van bovenstaande zijn.
<string const>	geeft een string met tekens aan, gesloten tussen aanhalingstekens, bijv. "MSX-COMPUTER".
<string var>	geeft een stringvariabele aan, zoals B$ of WOORD$.
<string exp>	geeft een uitdrukking aan die in een string kan resulteren, bijv. B$+"DE DAM","4",STR$(12).
<variabele>	kan zijn een 1) <string var>

	of, 2 <num var>.
<conditie>	geeft aan WAAR of NIET-WAAR-controle, zoals in A<0 of B$="DE DAM".
<opdracht>	een (of meerdere) BASIC-opdrachten.
<regel>	geeft het regelnummer aan.
<naam>	kan een programmanaam of functienaam zijn.

VOORBEELDEN

Dit gedeelte bevat enkele éénregelige voorbeelden en korte programma's die het gebruik van het sleutelwoord illustreren. Ook zal een korte uitleg van het voorbeeld worden gegeven.

Voor korte voorbeeldprogramma's worden de volgende symbolen gebruikt

---

```
Tussen deze lijnen staat een voorbeeld-
programma dat op uw computer kan worden gedraaid.
```

```
De resultaten van het voorbeeldprogramma worden
op deze manier weergegeven. Aanvullende informatie
wordt op dezelfde regel als de programmaregel gegeven.
```

PUNTEN OM TE ONTHOUDEN

Dit gedeelte geeft de mogelijkheden en beperkingen aan van het sleutelwoord. Er wordt een uitvoerige lijst van tips en adviezen gegeven, evenals aanvullende beschrijvingen van het sleutelwoord.

FOUTEN

Dit gedeelte geeft de mogelijke oorzaken van fouten die bij het sleutelwoord horen. Het is bedoeld om u te helpen bij het herstellen van fouten in uw programma.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Dit gedeelte geeft een overzicht van alle bijbehorende sleutelwoorden, die vaak samen met het sleutelwoord worden gebruikt.

Indien details van of een voorbeeld van een sleutelwoord ook in een ander gedeelte worden genoemd, zoals in het gedeelte over 'GEAVANCEERD PROGRAMMEREN', dan worden deze hoofdstukken aangegeven.

ABS ABSolute waarde

BESCHRIJVING
Deze functie geeft de absolute waarde terug, hetgeen het verschil is tussen het getal tussen haakjes en 0. Dit betekent in principe, dat negatieve getallen positief worden en positieve getallen ongewijzigd blijven. De absolute waarde van -2.6 is gelijk aan 2.6.
ABS wordt ook gebruikt om het verschil tussen twee waarden te berekenen.

SYNTAX

ABS (<num const>)
ABS (<num var>)
ABS (<num exp>)

VOORBEELDEN

```
10 PRINT "ABSOLUTE WAARDE VAN -1000 IS ";ABS(-1000)
20 A=1985
30 B=1960
40 PRINT ABS(B-A)
```

```
ABSOLUTE WAARDE VAN -1000 IS 1000
25
```

PUNTEN OM TE ONTHOUDEN
ABS is een functie waarvan het argument en het resultaat beide numeriek zijn. De ABS-waarde van een onbekende variabele is nul.

FOUTEN
Het door elkaar halen van een ABS-functie met een string resulteert in een 'Type Mismatch'-fout.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
SGN
Sectie 1, Hoofdstuk 15: Functies

AND Logische AND-bewerker

BESCHRIJVING

AND is één van de logische bewerkers die een meervoudige conditietest uitvoert met behulp van Boolean algebra. Boolean algebra vormt een belangrijke basis van de computerwetenschap. Voor meer details wordt verwezen naar het gedeelte over Boolean algebra.

AND wordt vaak gebruikt binnen IF...THEN...ELSE-opdrachten, om meer dan twee tests te doen, alvorens tot een resulterende actie wordt overgegaan. Bijvoorbeeld:

```
IF A=5 AND B$="ERNIE" THEN PRINT "BERT!" ELSE
PRINT "OK!"
```

De Boolean algebraïsche bewerking van AND kan als volgt in een waarheidstabel worden weergegeven.

AND	X	Y	X AND Y
	0	0	0
	1	0	0
	0	1	0
	1	1	1

Daarom kan 100 AND 50 als volgt worden uitgerekend:

	X	100	0000000001100100
AND	Y	50	0000000000110010
		32	0000000000100000

SYNTAX

IF <conditie> AND <conditie> ... THEN ... ELSE ...
<num var>=<num const> AND <num const>
<num var>=<num var> AND <num exp>

of andere numerieke combinaties.

VOORBEELDEN

Voorbeeld (1) van de IF...THEN...ELSE-conditietest:

```
10 T=12:HONGER$="ERGE"
20 INPUT "WILT U UW LUNCH (J/N)";AN$
30 IF T=12 AND HONGER$="ERGE" AND AN$="J" THEN PRINT
"NEEM WAT BOTERHAMMEN":END
40 PRINT "MOOI ZO! NEEM DAN LATER WAT."
```

Voer dit programma tweemaal uit. Zie wat er gebeurt als u 'J' ingeeft en als u 'N' ingeeft.

```
RUN
WILT U UW LUNCH (J/N)? J
NEEM WAT BOTERHAMMEN
OK
RUN
WILT U UW LUNCH (J/N)? N
MOOI ZO! NEEM DAN LATER WAT.
```

Voorbeeld (2) voor de Boolean algebra:

```
10 A%=185:B%=85
20 C%=A% AND B%
30 PRINT A%;" AND ";B%;"=";C%
40 PRINT "VERMENIGVULDIG DIT MET 100, DAN KRIJGT U";
100*(A% AND B%)
```

```
185 AND 85 = 17
VERMENIGVULDIG DIT MET 100, DAN KRIJGT U 1700
```

PUNTEN OM TE ONTHOUDEN

De logische AND kan worden gebruikt voor bittests van een bepaald getal. Bijvoorbeeld als $Y=2^n$, waarin n de test is die varieert tussen 0 en 7, dan zal X AND Y=Y zijn indien de testbit waar is (dus 1 is) en X AND Y=0 indien de testbit n niet waar is (dus 0 is).
BIT-TEST van de vijfde bit (n=5) voor X=117.

Y=2^5=32

	X	117	0000000001110101
AND	Y	32	0000000000100000
		32	0000000000100000

Daarom dus waar.
Boolean argumenten moeten liggen tussen de integere waarden -32768 tot 32767. Reële getallen worden afgekort tot integere waarden.

FOUTEN

Er treedt een 'Overflow Error' op als één van de argumenten een te grote waarde aanneemt, dus niet binnen de aangegeven marge valt (-32768 tot 32767).
Een 'Type Mismatch error' treedt op als één van de argumenten een string is.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

IF, THEN, ELSE, OR, EQV, XOR, NOT, IMP.
Sectie 1, Hoofdstuk 6: Het gebruik van condities
Sectie 2, Hoofdstuk 6: Boolean algebra
Sectie 2, Hoofdstuk 7: Boolean II: De IF...THEN...ELSE

ASC ASCII-code

BESCHRIJVING

Ieder teken in de computer heeft een corresponderende code, die ASCII-code wordt genoemd. (ASCII staat voor American Standard Code for Information Interchange.) De computer verandert alle strings in getallen omdat hij beter overweg kan met getallen dan met strings. De ASCII-code is bedoeld om verschillende computers die met ASCII werken dezelfde code te laten hebben voor een teken. Dit maakt het communiceren tussen computers onderling zeer eenvoudig.

Het kan voorkomen dat u wilt weten welke ASCII-code bij een bepaald teken hoort; ASC geeft u de ASCII-code van een bepaald teken, of van het eerste teken van een string.

SYNTAX

ASC ("<string>")	geeft de ASCII-code van het eerste teken in <string>. <string> kan langer zijn dan één teken.
ASC (<string var>)	geeft de ASCII-code van het eerste teken van de stringvariabele.

VOORBEELDEN

PRINT ASC("A")

resulteert in een afbeelding van de ASCII-code van 'A', en deze is 65.

```
10 A$="FANTASIE"
20 B=ASC(A$)
30 PRINT "ASCII-CODE VAN DE EERSTE LETTER VAN ";
A$;" IS ";B
```

Bovenstaand programma heeft als resultaat:

```
ASCII-CODE VAN DE EERSTE LETTER VAN FANTASIE
IS 70
```

PUNTEN OM TE ONTHOUDEN

Ieder teken, inclusief de besturingstekens, zoals <wagen-terug> en de grafische tekens hebben een eigen ASCII-code, en elke code kan worden bepaald met ASC.

Alleen het eerste teken van een string is van belang voor ASC. De resterende tekens worden genegeerd.

Het tegenovergestelde proces om een teken af te leiden van een ASCII-code wordt uitgevoerd door CHR$, doch dit geeft één teken per keer.

FOUTEN

Bevat een string niets, dus is het een nul-string, dan resulteert deze in de foutmelding 'Illegal Function Call'.
Bijvoorbeeld, 10 PRINT ASC(" ") geeft 'Illegal Function Call at 10'.
Is het argument numeriek, dan treedt een 'Type Mismatch' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

CHR$
Zie de Appendix voor een ASCII-tabel.
Sectie 1, Hoofdstuk 20: De ASCII-codes

ATN Arc-TaNgens

BESCHRIJVING
Deze functie geeft de arc-tangens van het argument in radialen tussen -PI/2 en PI/2.

SYNTAX

ATN(<num const>)
ATN(<num var>)
ATN(<num exp>)

ATN geeft de arc-tangens in radialen met dubbele nauwkeurigheid.

VOORBEELDEN

```
PRINT ATN(1.5)
 .98279372324731
PRINT ATN(3/6)
 .46364760900081
```

PI=4*ATN(1) ... 4*ATN(1) berekent de waarde van PI tot op 14 decimale posities.

PUNTEN OM TE ONTHOUDEN
ATN geeft altijd een dubbel nauwkeurig getal. Het argument kan iedere numerieke voorstelling zijn.

FOUTEN
Omdat het een trigonometrische functie betreft, zal het duidelijk zijn dat deze niet moet worden verward met strings. Gebeurt dit wel, dan treedt een 'Type Mismatch' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
TAN
SIN
COS
Sectie 1, Hoofdstuk 19: Wiskundige functies

AUTO AUTOmatische regelnummering

BESCHRIJVING

BASIC-programma's hebben een regelnummer nodig voor iedere programmaregel. Bij het intikken van een lang programma biedt AUTO u de mogelijkheid om automatisch regelnummers te genereren. Na iedere ingevoerde regel biedt de computer het volgende regelnummer aan (dus telkens na het intoetsen van RETURN).

Dit veraangenaamt het leven van de programmeur. De regelnummers hebben een constante verhogingsfactor.

SYNTAX

AUTO	geeft regelnummer vanaf 10 met een verhogingsfactor 10.
AUTO <num const>	regelnummers vanaf <num const> met verhogingsfactor 10.
AUTO <num const1>, <num const2>	regelnummers vanaf <num const1> met verhogingsfactor <num const2>.
AUTO ,<num const>	geeft regelnummers vanaf 10 met verhogingsfactor <num const>.
AUTO <num const>,	geeft regelnummers vanaf <num const> met dezelfde verhogingsfactor als hiervoor werd gebruikt.

VOORBEELDEN

AUTO geeft regelnummers 10, 20, 30, 40 enz.
AUTO 1000,5 geeft achtereenvolgens 1000, 1005, 1010 enz.

PUNTEN OM TE ONTHOUDEN

Nadat de <RETURN>-toets wordt ingedrukt biedt de computer u een nieuw regelnummer aan.

Er zijn twee manieren om de automatische regelnummering te onderbreken:

1. Druk tegelijkertijd <CTRL> en <STOP> in.
2. Druk tegelijkertijd <CTRL> en <C> in.

De regelnummerreeks dient te bestaan uit getallen die liggen tussen 0 en 65529 en de verhogingsfactor mag liggen tussen 1 en 65529. Wordt regelnummer 65529 bereikt, dan wordt vanzelf gestopt met automatische regelnummering en u wordt teruggeplaatst in de commando-mode.

Genereert de computer een regelnummer dat reeds bestaat, dan wordt een sterretje '*' geplaatst naast het regelnummer. Wilt u deze regel niet overschrijven, dan moet u <RETURN> indrukken. Wilt u de regel vervangen, negeer dan de '*'.

Functietoets F2 is geïnitialiseerd op 'AUTO' nadat de computer wordt aangeschakeld, zodat u met deze toets de automatische regelnummering kunt

aanroepen. De regelnummers dienen integere waarden te zijn.

FOUTEN

Is het opgegeven regelnummer of de verhogingsfactor geen integere waarde, dan wordt een 'Syntax error' gegeven.
Het is zeer eenvoudig om een gedeelte van een programma te verwijderen, zonder dat u die bedoeling heeft. Let daarom goed op de '*'-indicatie.
AUTO kan worden opgenomen in een BASIC-programma, maar dit is zinloos. Doe dit daarom niet.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

RENUM
Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's.
Sectie 2, Hoofdstuk 1: Geavanceerde programmabewerking.

BASE

BESCHRIJVING
Deze systeemvariabele geeft de beginpositie van de verschillende schermtabellen in de VIDEO-RAM. BASE kan worden behandeld als iedere andere variabele, maar zou alleen moeten worden gebruikt in geavanceerde grafische programma's. BASE is een 16-bit integere waarde.

SYNTAX
BASE(<numeriek>)

BASE(0)	-	beginadres van namentabel in tekstmodus 0.
BASE(1)	-	ongebruikt.
BASE(2)	-	beginadres van de patroongenerator in tekstmodus 1.
BASE(3)	-	ongebruikt.
BASE(4)	-	ongebruikt.
BASE(5)	-	beginadres van namentabel in tekstmodus 1.
BASE(6)	-	beginadres van kleurentabel in tekstmodus 1.
BASE(7)	-	beginadres van patroongenerator in tekstmodus 1.
BASE(8)	-	beginadres van sprite-attributen in tekstmodus 1.
BASE(9)	-	beginadres voor sprite-patroon in tekstmodus 1.
BASE(10)	-	beginadres van namentabel in grafische modus 2 met hoog oplossend vermogen.
BASE(11)	-	beginadres van kleurentabel in grafische modus 2 met hoog oplossend vermogen.
BASE(12)	-	beginadres van de patroongenerator in grafische modus 2 met hoog oplossend vermogen.
BASE(13)	-	beginadres van sprite-attributentabel in grafische modus 2 met hoog oplossend vermogen.
BASE(14)	-	beginadres van de sprite-patroontabel in grafische modus 2 met hoog oplossend vermogen.
BASE(15)	-	beginadres van namentabel in de grafische modus 3 met meer kleuren.

BASE(16) - ongebruikt.

BASE(17) - beginadres van de patroongeneratortabel in grafische modus 3 met meer kleuren.

BASE(18) - beginadres van de sprite-attributentabel in grafische modus 3 met meer kleuren.

BASE(19) - beginadres van de sprite-patroontabel in grafische modus 3 met meer kleuren.

VOORBEELDEN

```
PRINT BASE(0)
0
```

FOUTEN

Het argument mag alleen liggen tussen 0 en 19 en moet integer zijn. Andere waarden worden niet herkend en resulteren in een 'Illegal Function Call'-foutboodschap.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Sectie 2, Hoofdstuk 16: Geavanceerde grafische toepassingen VI: De VIDEO-RAM.

BEEP BEEP uit de luidspreker

BESCHRIJVING
BEEP geeft eenzelfde toon als in PRINT CHR$(7), 0.04 seconden lang.

SYNTAX
BEEP

VOORBEELDEN

```
BEEP
```

PUNTEN OM TE ONTHOUDEN
BEEP is gelijk aan CHR$(7).

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
SOUND
PLAY
Sectie 1, Hoofdstuk 28: De muziek-macrotaal

BIN$ BINaire string

BESCHRIJVING
BIN$ geeft de binaire representatie van een decimaal argument in een stringvorm. De waarde kan variëren tussen -32768 en 65535 en het argument moet integer zijn, of een numerieke uitdrukking met dezelfde condities.

SYNTAX

BIN$(<num const>)
BIN$(<num var>)
BIN$(<num exp>)

VOORBEELDEN

```
10 PRINT BIN$(255)
20 PRINT BIN$(9999)
```

```
RUN
11111111
10011100001111
```

PUNTEN OM TE ONTHOUDEN
Is het argument negatief dan wordt de twee-complementvorm gebruikt, dus BIN$(65536-n)
Om een binaire waarde te veranderen in een decimaal getal, kunt u &B als voorvoegsel gebruiken om het binaire getal voor te stellen en dient u deze gelijk te stellen aan een decimale variabele, zoals hieronder:

A%=&B00001111

FOUTEN
Het argument mag geen string of reëel getal zijn. In dit geval treedt er een 'Type Mismatch' op.
Er treedt een 'Overflow error' op als de integer buiten de toegestane reeks valt (-32678 tot 65535).

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 2, Hoofdstuk 5: Overzicht van getallensystemen gebruikt door MSX

BLOAD Laden van bytes

BESCHRIJVING

BLOAD wordt gebruikt om machinecode-programma's en gegevens (DATA) te laden van een opgegeven randapparaat, bijvoorbeeld een cassetterecorder. Er is een mogelijkheid om het programma automatisch uit te voeren, en ook om het op een geheugenplaats te laden die anders is dan die waarop het stond opgeslagen.

BLOAD wordt gebruikt voor het automatisch starten van machinecode-software. Voorheen kon alleen een cassetterecorder worden aangeroepen, maar tegenwoordig kan ook een micro floppy disk-drive worden aangeroepen.

SYNTAX

BLOAD "<randapparaat>" het laden van een bestand waarvan de naam niet bekend is.

BLOAD "<randapparaat>;<naam>" het laden van een bestand met de opgegeven naam.

BLOAD "<randapparaat>;<naam>",<num const> Met de opgegeven verplaatsing <num const> worden alle adressen zoals opgegeven in BSAVE verhoogd.

BLOAD "<randapparaat>;<naam>",R,<num const> laadt met de verplaatsing en voert deze vervolgens uit.

VOORBEELDEN

```
BLOAD "CAS:SPEL",R
BLOAD "CAS"AVONT",&HOOFF
```

PUNTEN OM TE ONTHOUDEN

Wordt het startadres weggelaten bij BSAVE, dan veronderstelt de computer dat het beginadres het startadres is zodra het programma wordt geladen met BLOAD en R-optie.

Veel van de machinecode-programma's zullen automatisch worden uitgevoerd met deze mogelijkheid.

De naam van het bestand mag maximaal zes tekens lang zijn.

FOUTEN

Er treedt een 'Device I/O error' op indien blijkt dat het bestand op de cassettetape is verminkt.

Geeft u een foutieve bestandsnaam op, dan zal duidelijk zijn dat de computer niets zal laden. Dit is een vaak voorkomende fout. Bent u niet zeker van de naam, laat deze dan weg en laadt aldus het eerste bestand dat aanwezig is.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

BSAVE

Sectie 2, Hoofdstuk 11: Het bewaren op en schrijven naar cassette

BSAVE Het bewaren van bytes

BESCHRIJVING
Dit commando schrijft een bepaalde hoeveelheid bytes vanuit een opgegeven geheugenpositie naar cassette of floppy disk. U moet hierbij het begin- en het eindadres opgeven. Dit commando wordt gebruikt om machinecode-programma's en gegevens (DATA) in de vorm van bytes te bewaren. U kunt ook het startadres meegeven, dat wordt aangeroepen bij de automatische uitvoering van het programma na laden met BLOAD.

SYNTAX
BSAVE "<randapparaat>;<naam>",<beginadres>,<eindadres>
BSAVE "<randapparaat>;<naam>",<beginadres>,<eindadres>,<startadres>

VOORBEELDEN
```
BSAVE "CAS:SPEL",&HA100,&HA200,&HCFFF
BSAVE "CAS:AVONT",&HC000,&HCFFF
```

PUNTEN OM TE ONTHOUDEN
De naam van het bestand mag uit maximaal zes tekens bestaan. Wordt het startadres weggelaten, dan veronderstelt de computer dat het beginadres het startadres is, indien wordt geladen met BLOAD en R-optie, om het programma automatisch uit te voeren.

FOUTEN
Optimaal opslaan van gegevens kan alleen worden bereikt met betrouwbare apparatuur. Dat is vanzelfsprekend. Lukt het wegschrijven niet, controleer dan steeds uw apparatuur.
Een bestandsnaam van meer dan zes tekens resulteert in een 'Syntax error'-boodschap.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
BLOAD
Sectie 2, Hoofdstuk 11: Het schrijven naar en lezen van cassette

CALL Roept een uitbreidingsopdracht aan

BESCHRIJVING

Deze opdracht roept een uitbreidingsopdracht aan die zich bevindt in een uitbreidings-ROM-cartridge. De afkorting voor CALL is '_', het onderstrepingssymbool. CALL is afhankelijk van de ROM-cartridge, zodat het gebruik ervan in een BASIC-programma betekent dat het programma niet meer binnen de MSX-standaard valt.

SYNTAX

CALL <naam>(<lijst met argumenten>)
_<naam>(<lijst met argumenten>)
De lijst met argumenten kan van alles zijn. Het hangt er nu eenmaal van af wat de uitbreidingsopdracht doet.

VOORBEELDEN

```
CALL SPEECH (volume%,stem%,"HALLO")
```

PUNTEN OM TE ONTHOUDEN

Er zijn computers die CALL gebruiken om een machinecode-routine aan te roepen, doch dit is niet het geval bij MSX. Om dit te bereiken moet u USR gebruiken.
De maximum lengte van de uitbreidingsopdracht is 15 tekens. Tenzij u een uitbreidings-ROM heeft aangesloten, is deze functie zinloos. Wat een CALL gaat uitvoeren hangt af van de ROM.

FOUTEN

Gebruik CALL niet voor commerciële software, omdat het niet past binnen de MSX-standaard.
Een niet bestaande naam van een CALL resulteert in een 'Syntax error'-boodschap.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Sectie 2, Hoofdstuk 22: MSX-geheugenorganisatie en cartridge-slotmechanisme

CDBL Veranderen in een dubbel nauwkeurig getal

BESCHRIJVING
De CDBL-functie verandert integere en enkelvoudig nauwkeurige getallen, variabelen en uitdrukkingen in dubbel nauwkeurige getallen.

SYNTAX
CDBL(<num const>)
CDBL(<num var>)
CDBL(<num exp>)

VOORBEELDEN
```
B#-CDBL(A!)
PRINT CDBL(NO%)
```

PUNTEN OM TE ONTHOUDEN
Dubbel nauwkeurige getallen zijn 8 bytes lang en ze zijn tot op 14 decimale posities nauwkeurig.

FOUTEN
Wordt een string als argument gebruikt dan treedt een 'Type Mismatch' op, of als CDBL wordt gelijkgesteld aan een stringvariabele.

BIJBEHORENDE SLEUTELWOORDEN OF VERWIJZINGEN
CINT
CSGN
Sectie 2, Hoofdstuk 3: Soortverandering

CHR$ Karakterstring

BESCHRIJVING

CHR$ genereert een karakter of teken van een opgegeven getal dat correspondeert met de ASCII-code. Dit is dus de complementfunctie van ASC.
Omdat enkele van de ASCII-codes corresponderen met besturingstekens, zoals het schoonmaken van het beeldscherm, kunt u deze activeren met PRINT CHR$.

SYNTAX

CHR$(<num const>)
CHR$(<num var>)
CHR$(<num exp>)
Het bovenstaande resulteert in een karakterstring.

VOORBEELDEN

PRINT CHR$(66) drukt het teken 'B' af op het scherm.

```
10 C1=65:REM ASCII-code van A is 65
20 C2=72:REM ASCII-code van H is 72
30 C3=84:REM ASCII-code van T is 84
40 PRINT CHR$(C2);CHR$(C1);CHR$(C3)
50 A$=CHR$(C2+32)+CHR$(C1+32)+CHR$(C3+32)
60 PRINT A$
```

Als u het bovenstaande uitvoert, zult u merken dat, wanneer u 32 toevoegt aan de ASCII-code van de hoofdletters, u kleine letters krijgt.

```
RUN
HAT
hat
```

PUNTEN OM TE ONTHOUDEN

Als het getal tussen haakjes ligt tussen 0 en 31, zal CHR$ reageren zoals aangegeven door het besturingsteken in de ASCII-standaard. Bijvoorbeeld PRINT CHR$(7) geeft een BEEP uit de luidspreker. Kijk eens in de besturingstekentabel. U zult merken dat u verschillende besturingstekens kunt opnemen in één string.
Ligt het getal tussen haakjes tussen 32 en 255, dan geeft CHR$ het corresponderende ASCII-teken.

FOUTEN

Is het getal tussen haakjes minder dan nul, of meer dan 255, dan treedt een 'Illegal Function Call' op.
Onthoud goed dat CHR$ een string voorstelt en daarom niet kan worden

gelijkgesteld aan een numerieke variabele. Als u dit wel doet treedt er een 'Type Mismatch' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ASC
Zie Appendix voor de ASCII-karaktercodetabel.
Sectie 2, Hoofdstuk 20: De ASCII-codes

CINT Verander in integer

BESCHRIJVING

Deze functie wijzigt enkelvoudige en dubbel nauwkeurige getallen, variabelen en uitdrukkingen in integere waarden.
De waarde mag liggen tussen -32768 en 32767. Alles wat hier buiten valt veroorzaakt problemen.
Breuken van een reëel getal zullen worden afgekort.

SYNTAX

CINT(<num const>)
CINT(<num var>)
CINT(<num exp>)

VOORBEELDEN

```
A%=CINT(B!*C#)
PRINT CINT(1234.56789)
```
geeft 1234

PUNTEN OM TE ONTHOUDEN

Integere waarden zijn twee bytes groot.

FOUTEN

Het argument moet liggen tussen -32768 en 32767. Anders treedt er een 'Overflow error' op.
Wordt een string als argument gebruikt dan treedt er een 'Type Mismatch' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

CDBL
CSNG
INT
FIX
Sectie 2, Hoofdstuk 3: Soortverandering

CIRCLE   Teken een cirkel

BESCHRIJVING
CIRCLE tekent een hele of een deel van een cirkel of een ellips in de grafische modus. U kunt opgeven wat het middelpunt is, wat de kleur moet zijn en zelfs welk gedeelte van de cirkel u door de computer wilt laten tekenen.

SYNTAX
CIRCLE <coördinaten>,<straal>,<kleur>,<starthoek>,<eindhoek>,<aspect ratio>
Elk van deze parameters kan zijn: <num const>, <num var> of <num exp> als ze maar liggen in de toegestane reeks (zie hieronder).

Verklaring van <coördinaten>
Er zijn twee manieren om de coördinaten van het middelpunt aan te geven, de ene is absoluut en de andere is relatief.
<coördinaten> kan zijn:

1. (<x-coördinaat>,<y-coördinaat>)
   Deze coördinaten geven de absolute positie van het middelpunt.
2. STEP (<x-offset>,<y-offset>)
   Deze coördinaten zijn relatief ten opzichte van het laatste punt waar de computer naar heeft gerefereerd. Valt het resultaat hierdoor buiten het beeldscherm, dan tekent de computer tot aan de rand van het scherm.

<kleur>,<starthoek>,<eindhoek> en <aspect ratio> zijn optioneel.

Reeks:

<x-coördinaat>	-32768 tot 32767. Moet het middelpunt binnen het beeldscherm vallen, gebruik dan 0 tot 255.
<y-coördinaat>	-32768 tot 32767. Moet het middelpunt binnen het beeldscherm vallen, gebruik dan 0 tot 191.
<straal>	0 tot 32767, maar de cirkel past niet op het scherm als de straal te groot wordt gekozen.
<kleur>	0 tot 15; zie ook de kleurcode in de sectie over COLOR.
<starthoek>	0 tot 2*PI in radialen.
<beginhoek>	0 tot 2*PI in radialen.

Een minteken ('-') als voorvoegsel voor <starthoek> en <eindhoek> geeft aan dat een lijn moet worden getrokken van het centrum naar ieder eindpunt.

<aspect ratio>	0 tot 32767.
<aspect ratio>=	(verticale straal)/(horizontale straal).

Is de <aspect ratio> minder dan 1, dan veronderstelt de computer deze als gelijk aan 1 en tekent dus een cirkel.

Is <aspect ratio> erg groot, dan krijgt u een verticale lijn. Is de <aspect ratio> 0, dan krijgt u een horizontale lijn.

VOORBEELDEN

```
10 SCREEN 2
20 CIRCLE (100,100),50,8
30 FOR I=1 TO 5
40 CIRCLE STEP (I*5,0),50,8
50 NEXT
60 GOTO 60:REM DIT BEVRIEST HET SCHERM
```

PUNTEN OM TE ONTHOUDEN

Dit commando werkt alleen in de grafische modus, overtuig u daarom eerst dat u in SCREEN 2- of SCREEN 3-modus bent.

Is de kleur niet opgegeven, dan gebruikt de computer de huidige voorgrondkleur.

De begin- en eindhoek worden opgegeven in radialen.

FOUTEN

Een 'Overflow error' treedt op als de waarde van <coördinaten> of <straal> buiten de toegestane reeks valt.

Een 'Illegal Function Call' treedt op bij een incorrect opgegeven kleurcode. U krijgt deze boodschap ook als de <beginhoek> en de <eindhoek> buiten de toegestane reeks vallen.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

COLOR

SCREEN

Sectie 1, Hoofdstuk 25: Het tekenen van cirkels en ellipsen.

CLEAR Wis een geheugengebied

BESCHRIJVING

CLEAR geeft de computer aan dat hij alle huidige variabelen die in het geheugen aanwezig zijn moet vergeten. Hierbij worden alle strings en numerieke variabelen buiten werking gesteld.
U kunt met dit commando ook een geheugengebied reserveren voor string-variabelen en machinecoderoutines in het systeem-werkgebied.
U kunt geheugen reserveren door het wijzigen van de hoogste geheugenlocatie die beschikbaar is voor BASIC. De vervangingswaarde voor de hoogste geheugenpositie is &HF380. Deze waarde kan worden verlaagd met CLEAR, <hoogste geh>. Het aldus verkregen gebied kan worden gebruikt om machine-coderoutines en gegevens in op te slaan.

SYNTAX

CLEAR <string-ruimte>	bepaalt de grootte van het string-gebied.
CLEAR,<hoogste geh>	bepaalt de hoogste geheugenpositie die beschikbaar is voor BASIC.
CLEAR <string-ruimte>, <hoogste geh>	bepaalt zowel de grootte van het string-gebied, als de hoogste geheugenpositie die beschikbaar is voor BASIC.

VOORBEELDEN

`CLEAR`	
`CLEAR 255`	verhoogt het string-gebied met 255 bytes.
`CLEAR,&HF300`	creëert geheugenruimte voor machinecode-programma's tussen &HF300 en &HF380.

PUNTEN OM TE ONTHOUDEN

De initiële stringruimte is 200 bytes. Dit zet de maximum lengte van een stringvariabele op 200 tekens. Wilt u dat de computer meer dan 200 tekens toestaat voor een variabele, dan moet u eerst een CLEAR 255-opdracht uitvoeren.
Het geheugengebied tussen &HFFFF en &HF380 wordt gebruikt door de BASIC-ROM als systeem-werkgebied.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's
Sectie 2, Hoofdstuk 2: Constanten en variabelen
Sectie 2, Hoofdstuk 20: Geheugenorganisatie

CLOAD/CLOAD? Laden van cassette

BESCHRIJVING

Gebruik CLOAD om een BASIC-programma van cassette te laden. De computer geeft aan of hij het programma heeft gevonden en of hij dit overslaat of laadt. U hoeft de naam van het programma niet te weten. Als u geen naam meegeeft, laadt CLOAD het eerste programma dat hij tegenkomt op de tape.
Of een programma succesvol wordt geladen of niet, hangt eigenlijk alleen maar af van de kwaliteit van de cassetterecorder.
CLOAD? (CLOAD met vraagteken) verifieert een BASIC-programma dat zojuist is geladen van cassette met een versie in het computergeheugen. De computer geeft 'OK' of een verificatiefout als er een verschil blijkt te zijn.

SYNTAX

CLOAD
CLOAD"<naam>"
CLOAD?
CLOAD?"<naam>"

VOORBEELDEN

Gesteld, we hebben twee programma's, namelijk 'SLANG' en 'LADDER' op de cassette. Als u dan intikt:

CLOAD <RETURN>

```
CLOAD
Found: SLANG
OK
```

Probeer eens CLOAD "LADDER" <RETURN>.

```
CLOAD "LADDER"
Skip: SLANG
Found: LADDER
OK
```

PUNTEN OM TE ONTHOUDEN

De programmanaam mag niet langer dan zes tekens zijn. Wordt een programma geladen met CLOAD, dan wordt het programma dat op dat moment aanwezig is in het computergeheugen gewist.
De functietoets F7 is CLOAD".
De opneemsnelheid kan zowel 1200 BAUD als 2400 BAUD zijn, maar CLOAD zal

automatisch registreren welke snelheid van toepassing is en dienovereenkomstig laden.
Is er sprake van een geopend bestand (met OPEN), dan zal CLOAD dit automatisch sluiten en vervolgens weer doorgaan.

FOUTEN
Een 'Device I/O error' treedt op als u een incorrect opgenomen programma van cassette probeert te laden. Gebruik daarom altijd een betrouwbare cassetterecorder.
Geeft u een foutieve bestandsnaam op dan zal duidelijk zijn dat er niets wordt geladen. Let op uw spelling.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
CSAVE
Sectie 1, Hoofdstuk 11: Het bewaren van uw programma's op tape

CLOSE Sluit een kanaal (channel)

BESCHRIJVING
Dit commando wordt gebruikt om een geopend kanaal te sluiten. Het is het tegenovergestelde van OPEN. De bijbehorende buffer van dit kanaal wordt ook vrijgegeven.

SYNTAX
CLOSE ... sluit alles.
CLOSE#<bestandsnummer>,<bestandsnummer>

VOORBEELDEN
```
CLOSE#2
```

FOUTEN
Er volgt een 'Bad File Number' indien een niet-bestaand bestandsnummer wordt meegegeven.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
OPEN
Sectie 2, Hoofdstuk 19: Hoe bestanden worden gebruikt

CLS Wist het scherm

BESCHRIJVING
Wist het scherm in iedere modus. In de grafische modus wordt het scherm van de huidige achtergrondkleur voorzien.

SYNTAX
CLS

VOORBEELDEN
Het volgende wist het scherm en verandert dit in helderrood:

```
10 SCREEN 2
20 COLOR ,9
30 CLS
40 GOTO 40
```

REGEL 10 selecteert HIRES-modus.
REGEL 20 achtergrondkleur op 9 (helderrood).
REGEL 30 wist het scherm.
REGEL 40 bevriest het scherm.
Resultaat: een helderrood scherm.

PUNTEN OM TE ONTHOUDEN
De tekstcursor wordt links bovenaan het scherm geplaatst. Indien de functietoetsen in KEY ON-modus zijn, blijft het overzicht van functietoetsen onderaan het scherm zichtbaar, ook na het wissen. Dit overzicht verdwijnt pas na KEY OFF.
<CTRL><L> heeft dezelfde functie als CLS.
PRINT CHR$(12) heeft dezelfde functie als CLS, dus u kunt een CLS in een string invoegen.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
COLOR
Sectie 1, Hoofdstuk 3: Het schrijven van een programma

COLOR Kleur

BESCHRIJVING

Deze opdracht bepaalt de achtergrond-, voorgrond- en randkleur. Er zijn 15 kleuren en een transparant.

SYNTAX

COLOR<voorgrondkleur>,<achtergrondkleur>,<randkleur>

De argumenten kunnen allemaal <num exp> zijn, maar moeten liggen tussen 0 en 15. Ze hoeven niet allemaal te worden gespecificeerd.

KLEURENCODE

0 transparant	8 middelrood
1 zwart	9 lichtrood
2 middelgroen	10 donkergeel
3 lichtgroen	11 lichtgeel
4 donkerblauw	12 donkergroen
5 lichtblauw	13 paarsrood
6 donkerrood	14 grijs
7 cyaanblauw	15 wit

VOORBEELDEN

`COLOR 11`	zet de voorgrondkleur op lichtgeel.
`COLOR 15,15`	zet de randkleur en de voorgrondkleur op wit.
`COLOR 1,5,1`	zet de achtergrondkleur op lichtblauw en de rand- en de voorgrondkleur op zwart.

PUNTEN OM TE ONTHOUDEN

Nadat de machine wordt aangeschakeld worden de kleuren volgens COLOR 15,4,7 opgezet, dus een witte voorgrond, een lichtblauwe achtergrond en een cyaanblauwe rand.

Functietoets F1 wordt voorgedefinieerd op 'COLOR'. Functietoets F6 wordt op 'COLOR 15,4,7' gezet, welke de vervangingskleuren voorstellen.

De achtergrondkleur zal niet veranderen, tenzij een CLS wordt uitgevoerd.

Indien de kleur niet is aangegeven in PSET, LINE of CIRCLE, tekenen deze grafische opdrachten in de kleuren zoals aangegeven met COLOR.

FOUTEN

De argumenten moeten liggen tussen 0 en 15; elke andere waarde resulteert in een 'Illegal Function Call'-boodschap.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

DRAW, PRESET, PSET, LINE, CIRCLE, CLS

Sectie 1, Hoofdstuk 22: COLOR

Sectie 2, Hoofdstuk 13: Geavanceerde grafische toepassingen II: kleuren in de HIRES grafische modus 2

CONT CONTinue

BESCHRIJVING
Deze opdracht geeft de computer aan, de uitvoering van een programma voort te zetten vanaf een punt waar deze het laatst werd onderbroken door een STOP-opdracht of <CTRL> <STOP>. De computer onthoudt de laatst uitgevoerde regel voordat <CTRL> <STOP> werd ingedrukt en zet de uitvoering voort vanaf de daaropvolgende regel. Trad de onderbreking op in het midden van een INPUT-opdracht dan begint de computer opnieuw bij INPUT.

SYNTAX
CONT

VOORBEELDEN

```
10 PRINT "GEEF UW GEBOORTEDATUM"
20 INPUT DAG,MND,JAAR
30 PRINT "DANK U"
```

```
RUN
GEEF UW GEBOORTEDATUM
?1970
??13          DRUK <CTRL> <STOP> IN
BREAK IN 20
OK
CONT
?1970
??12
??1
DANK U
OK
```

PUNTEN OM TE ONTHOUDEN
CONT reageert in de volgende situaties:

1. Nadat een programma werd gestopt met STOP. Het programma zet de uitvoering voort vanaf de volgende regel.
2. Nadat het programma is gestopt door een END. Het programma gaat door vanaf de volgende regel.
3. Nadat het programma werd gestopt door <CTRL> <STOP>. Het programma gaat door vanaf de volgende regel.

FOUTEN
CONT werkt niet in de volgende situaties:

1. Nadat het programma werd bewerkt.
2. Na het stoppen van de printer.
3. Na het beëindigen van invoer- of uitvoer-opdrachten van de cassette, zoals INPUT#.

In alle genoemde gevallen volgt de boodschap 'Can't Continue'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's

COS COSinus

BESCHRIJVING
COS geeft de cosinus van een bepaalde hoek. COS wordt uitgerekend in dubbele nauwkeurigheid. Het argument, dus de hoek, moet in radialen worden gegeven. MSX herkent geen graden.

SYNTAX
COS(<num const>)
COS(<num var>)
COS(<num uitdrukking>)

VOORBEELDEN
```
PRINT COS(1.4445432)
1259179846524
```

PUNTEN OM TE ONTHOUDEN
COS met dubbele nauwkeurigheid.
Het argument moet in radialen worden gegeven.

FOUTEN
Dit is een trigonometrische functie en daarom worden in deze functie geen strings geaccepteerd. In geval van bijvoorbeeld COS(A$) treedt de foutmelding 'Type Mismatch' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
SIN
TAN
ATN
Sectie 1, Hoofdstuk 19: Wiskundige functies

CSAVE SAVE naar cassette

BESCHRIJVING

CSAVE wordt gebruikt om een BASIC-programma op een cassettetape te bewaren. Het bewaarde programma moet een bestandsnaam hebben van zes tekens of minder. CSAVE schrijft het BASIC-programma weg in symbolen, in plaats van als ASCII-bestand, om zodoende tijd te besparen. De BAUD-waarde kan 1200 of 2400 zijn, afhankelijk van wat u aangeeft. Geeft u geen BAUD-waarde aan, dan veronderstelt de computer 1200 BAUD.

SYNTAX

CSAVE "<naam>"
CSAVE "<naam>",<BAUD-waarde>
<BAUD-waarde> : 1 = 1200 BAUD, 2 = 2400 BAUD

VOORBEELDEN

```
CSAVE "AVONTUUR"
CSAVE "SPEL",2
```

PUNTEN OM TE ONTHOUDEN

De naam van het programma mag zes tekens of minder zijn.
De BAUD-waarde kan ook worden bepaald via SCREEN.
Ieder programma dat met CSAVE is weggeschreven kan niet met een MERGE-opdracht worden samengevoegd met een ander programma vanwege de opslag in symbolen. Om programma's samen te voegen dient u de programma's als ASCII-bestanden op te slaan met behulp van SAVE.

FOUTEN

Het succes van CSAVE hangt voornamelijk af van de kwaliteit van de cassettetape en de cassetterecorder. Een slechte opname resulteert in een 'Device I/O error' als u weer probeert op te nemen met bijv. CLOAD.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

CLOAD
SAVE
Sectie 1, Hoofdstuk 11: Het bewaren van programma's op tape

CSNG Veranderen in een enkelvoudig nauwkeurig getal

BESCHRIJVING
Deze functie verandert het argument in een getal van enkelvoudige nauwkeurigheid.

SYNTAX
CSNG(<num const>)
CSNG(<num var>)
CSNG(<num exp>)

VOORBEELDEN
```
PRINT COS(0.7656)
 .72096670541357
PRINT CSNG(COS(0.7656))
 .720967
```

PUNTEN OM TE ONTHOUDEN
CSNG rondt af tot op zes decimale posities.

FOUTEN
Wordt een string als argument meegegeven dan treedt er een 'Type Mismatch' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
CDBL
CINT
Sectie 2, Hoofdstuk 3: Soortverandering

CSRLIN Cursor-regelpositie

BESCHRIJVING
Deze systeemvariabele geeft u de positie van de tekstcursor.

SYNTAX
CSRLIN

VOORBEELDEN
```
PRINT CSRLIN
```

PUNTEN OM TE ONTHOUDEN
CSRLIN is 0 (nul) voor de topregel.

FOUTEN
U kunt deze variabele niet toekennen. Er treedt een 'Syntax error' op als u dit toch probeert.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
POS(0)
Sectie 2, Hoofdstuk 12: Geavanceerde grafische toepassingen I: Karakteristieken van iedere schermmodus.

DATA

BESCHRIJVING

In DATA-opdrachten kunt u gegevens opslaan die nodig zijn in een BASIC-programma, die dan kunnen worden gelezen met READ-opdrachten. U kunt iedere willekeurige hoeveelheid getallen of strings opslaan die op een programmaregel passen. Ieder element dient te worden gescheiden met behulp van een komma. De plaats van DATA-regels is onbelangrijk, omdat ze nooit worden uitgevoerd; de computer negeert DATA-regels zodra hij deze tegenkomt; zij kunnen daarom overal worden geplaatst.
De READ-opdracht wordt gebruikt om achtereenvolgens de elementen te lezen die staan in de DATA-regels. De computer onthoudt welke van de elementen reeds zijn gelezen. Iedere volgende READ-opdracht begint bij het eerstvolgende ongelezen element.
De elementen in een DATA-regel kunnen zowel getallen als strings zijn. De lengte van de lijst mag echter hoogstens 255 tekens zijn, om in een regel te passen. De numerieke constante kan een integer, enkel- of dubbelnauwkeurig getal zijn. De stringconstanten hoeven niet tussen aanhalingstekens te staan, tenzij de string komma's, dubbele punten of puntkomma's enz. bevat. U kunt geen variabelen of uitdrukkingen opnemen in DATA-regels.
De corresponderende variabele in een READ-opdracht moet van dezelfde soort zijn als in de DATA-regel, dit is een stringvariabele voor stringgegevens.

SYNTAX

DATA <lijst met constanten>

VOORBEELDEN

```
10 PRINT "LIJST VAN TELEFOONNUMMERS"
20 FOR I=1 TO 4
30 READ NME$,PSTC$,TEL%
40 PRINT NME$,PSTC$,;"-";TEL%
50 NEXT I
60 DATA TOM,123,12345,PAUL,999,22222
70 DATA ELS,555,11111,JEAN,456,10001
```

```
RUN
LIJST VAN TELEFOONNUMMERS
TOM           123-12345
PAUL          999-22222
ELS           555-11111
JEAN          456-10001
```

PUNTEN OM TE ONTHOUDEN

Aanhalingstekens rond stringgegevens zijn nodig in de volgende gevallen:

1. Indien de stringgegevens komma's bevatten.
2. Indien de stringgegevens dubbele punten bevatten.
3. Indien de stringgegevens spaties bevatten voor of achter de stringtekens.

Om naar een specifieke DATA-regel te refereren kunt u RESTORE gebruiken, opdat de computer bij een READ-opdracht vanaf de aangegeven regel verder leest.

FOUTEN

Verschillen de tekensoorten in DATA- en READ-opdracht dan treedt een 'Type Mismatch' op. Daarom moeten stringvariabelen stringgegevens lezen en numerieke variabelen getallen.
Als de computer geen elementen meer tegenkomt bij een READ/DATA-opdracht dan treedt een 'Out of DATA'-fout op

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

READ
RESTORE
Sectie 1, Hoofdstuk 12: Het lezen van gegevens in arrays

DEFDBL Definieer dubbele nauwkeurigheid

BESCHRIJVING
De DEFDBL-opdracht verklaart dat iedere variabele die begint met de opgegeven reeks letters wordt behandeld als een getal met dubbele nauwkeurigheid.

SYNTAX
DEFBDL <reeks letters>

VOORBEELDEN
`5 DEFBDL A,B,C, ...` dit is dubbele nauwkeurigheid voor de variabelen die beginnen met A, B en C.

PUNTEN OM TE ONTHOUDEN
De verklaring van de variabelesoort met #, $ en ! hebben prioriteit boven DEFBDL.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
DEFINT
DEFSNG
DEFSTR
Sectie 2, Hoofdstuk 2: Constanten en variabelen

DEF FN Definieer een functie

BESCHRIJVING
Met behulp van DEF FN kunt u uw eigen functies definiëren, die met een willekeurig gekozen naam kunnen worden aangeroepen in uw programma. DEF FN is ontworpen om geheugen te besparen bij uitdrukkingen die meerdere keren nodig zijn in een programma.
U kunt de functie iedere willekeurige naam meegeven, zolang deze naam maar niet bestaat uit gereserveerde woorden.
De DEF FN-functie moet worden uitgevoerd om de computer te laten weten dat de functie bestaat, voordat deze wordt aangeroepen.
U kunt parameters aan de functie meegeven. Iedere variabele in de functie is plaatselijk, dus de waarde ervan blijft ongewijzigd buiten de functie.
Om een functie aan te roepen moet u FN plaatsen voor de naam van de functie.
De DEF FN-functie is beperkt tot één regel.

SYNTAX
DEF FN<naam>=<uitdrukking>
DEF FN<naam>(<parameters>)=<uitdrukking>
<naam> moet een $ (dollarteken) als achtervoegsel hebben vanwege de stringfunctie.
Indien gewenst mogen andere soorten specificaties worden gebruikt.

VOORBEELDEN

```
10 DEF FNCUBE(X)=X^3
20 INPUT "GEEF EEN GETAL OP";A%
30 PRINT "DE DERDE MACHT VAN ";A%;"IS";FNCUBE(A%)
40 B%=FNCUBE(A%+1)
50 PRINT "DE DERDE MACHT VAN ";A%+1;"IS";B%
60 C%=FNCUBE(FNCUBE(A%))
70 PRINT "DE DERDE MACHT VAN ";FNCUBE(A%);"IS";C%
```

```
RUN
GEEF EEN GETAL OP? 3
DE DERDE MACHT VAN 3 IS 27
DE DERDE MACHT VAN 4 IS 64
DE DERDE MACHT VAN 27 IS 19683
```

PUNTEN OM TE ONTHOUDEN
De elementen in de parameterlijst, zoals opgegeven in de definitie van de functie, kunnen zowel numeriek, strings, uitdrukkingen of variabelen zijn.
U kunt ook variabelen opnemen die worden gebruikt buiten de functie.

U kunt een geneste functie opzetten, door een functie op te nemen in een parameter, zoals in het hiervoor gegeven voorbeeld.

FOUTEN

De meeste fouten die voorkomen bij het gebruik van DEF FN ontstaan bij het aanroepen van de functie:

1. Ongedefinieerde gebruikersfunctie, indien een functie werd aangeroepen voordat DEF FN werd uitgevoerd.
2. Indien de variabelesoort niet past bij de aan de functie toegekende variabele en het resultaat van de functie, treedt een 'Type Mismatch'-foutmelding op; bijv. A$=FNCUBE(2).
3. Zijn er onvoldoende parameters meegegeven bij de aanroep van de functie dan treedt een fout op.
4. Een 'Type Mismatch'-foutmelding treedt op indien de parametersoorten niet passen.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Sectie 1, Hoofdstuk 16: Het definiëren van uw eigen functies

DEFINT Definieer integer

BESCHRIJVING
De DEFINT-opdracht verklaart dat iedere naam van een variabele die begint met de aangegeven reeks tekens, dient te worden behandeld als integere waarde.

SYNTAX
DEFINT <reeks tekens>

VOORBEELD
`10 DEFINT I,J,K ...` variabelen die beginnen met I, J en K zijn integere waarden.

PUNTEN OM TE ONTHOUDEN
De verklaring van variabelensoorten door middel van #, $, % en ! hebben prioriteit boven DEFINT.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
DEFDBL
DEFSGN
DEFSTR
Sectie 2, Hoofdstuk 2: Constanten en variabelen

DEFSGN Definieer enkelvoudige nauwkeurigheid

BESCHRIJVING
De DEFSGN-opdracht verklaart dat iedere variabele die begint met de opgegeven reeks letters moet worden beschouwd als een variabele met enkelvoudige nauwkeurigheid.

SYNTAX
DEFSGN <reeks letters>

VOORBEELDEN
`10 DEFSGN X,Y,Z ...` enkelvoudige nauwkeurigheid voor variabelen die beginnen met X, Y en Z.

PUNTEN OM TE ONTHOUDEN
De verklaring van de variabelensoort met #, $, % en ! heeft prioriteit boven DEFSGN.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
DEFDBL
DEFINT
DEFSTR
Sectie 2, Hoofdstuk 2: Constanten en variabelen

DEFSTR Definieer een string

BESCHRIJVING
De DEFSTR-opdracht verklaart dat iedere variabele waarvan de naam begint met één van de opgegeven reeks letters dient te worden beschouwd als stringvariabele.

SYNTAX
DEFSTR <reeks letters>

VOORBEELDEN
```
DEFSTR E,R,T
```

PUNTEN OM TE ONTHOUDEN
De verklaring van de variabelensoort met #, $, % en ! heeft prioriteit boven DEFSTR.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
DEFDBL
DEFINT
DEFSNG
Sectie 2, Hoofdstuk 2: Constanten en variabelen

DEF USR — Definieer een machinecoderoutine van de gebruiker

BESCHRIJVING

Om machinecodesubroutines te gebruiken vanuit BASIC kunt u de USR-functie gebruiken. De USR-functie roept een machinecodeprogramma aan op het adres, zoals aangegeven in de DEF USR-opdracht.
U kunt tot 10 USR-functies definiëren. Iedere USR moet worden gevolgd door een cijfer, hetgeen 5 is in onderstaand voorbeeld:

```
DEF USR5=&HFF80
```

Wilt u de machinecoderoutine aanroepen die is opgegeven in de DEF USR, gebruik dan USR, dus:

```
PRINT USR5("parameter")
```

waarin de parameter tussen de haakjes de parameter is die wordt overgedragen vanuit BASIC naar de machinecode.

SYNTAX

DEF USR=<startadres> ... als [cijfer]=0
DEF USR[cijfer]=<startadres>
<startadres> kan zowel <num const>, <num var> of <numerieke uitdrukking> zijn, maar moet in ieder geval een integer zijn.
[cijfer] moet liggen tussen 0 en 9.

VOORBEELDEN

```
DEF USR5=&HF300
DEF USR=&HF300+255
```

Om USR aan te roepen:

```
10 DUMMY=USR5(9)
```

(9) is de parameter die wordt overgedragen aan de machinecode.

PUNTEN OM TE ONTHOUDEN

U kunt hooguit 10 USR's aanroepen, maar ze kunnen opnieuw worden gedefinieerd, zodat u net zoveel routines kunt gebruiken als nodig is.
DEF USR is identiek aan DEF USR0.
U dient over machinecodekennis te beschikken van de Z80, om DEF USR en USR correct te kunnen gebruiken.

FOUTEN

Het startadres dient binnen de RAM te liggen en moet integer zijn. Ieder ander adres resulteert in een foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Sectie 2, Hoofdstuk 21: USR-functie en machinecode

DELETE

BESCHRIJVING
Dit is een bewerkingscommando dat een gedeelte van het programma schrapt.

SYNTAX
DELETE <regel>
DELETE <regel>-<regel>
DELETE -<regel>

VOORBEELDEN

`DELETE 20`	schrapt regel 20
`DELETE 20-100`	schrapt regels 20 tot en met 100
`DELETE -100`	schrapt regels van het begin tot en met 100

PUNTEN OM TE ONTHOUDEN
Bij het schrappen van slechts één regel is het veel eenvoudiger om het regelnummer in te tikken en daarna <RETURN>, in plaats van het DELETE-commando hiervoor te gebruiken.
DELETE kan niet in een programma worden gebruikt.

FOUTEN
Verwijst u naar een niet-bestaand regelnummer dan treedt een 'Illegal Function Call'-foutmelding op.
Schrapt u per ongeluk een gedeelte van een programma dan krijgt u een grote warboel. Wees er altijd zeker van dat u geen noodzakelijke regels verwijdert.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's
Sectie 2, Hoofdstuk 1: Geavanceerde programmabewerking

DIM Dimensionale array

BESCHRIJVING

Het DIM-commando wordt gebruikt om arrays op te zetten. Een array is een groep variabelen die alle dezelfde naam hebben maar van elkaar verschillen door een indexnummer.

SYNTAX

DIM <variabele>(<num const>)

DIM <variabele>(<num const>,<num const>, ...)

<variabele> is de naam van de array die wordt opgezet en kan integer, enkelvoudig nauwkeurig, dubbel nauwkeurig of een string zijn.

<num const> geeft de grootte van de array aan en moet een positieve integer zijn. Om een meervoudig dimensionale array op te zetten, plaatst u meer dan één <num const> tussen de haakjes.

VOORBEELDEN

```
DIM A(5,5,5)     is een drie-dimensionale array
DIM B$(100),C$(100,2),D%(100),E!(100),F#(100)
```

U kunt in een DIM-opdracht elke soort array definiëren en net zoveel als u nodig heeft.

PUNTEN OM TE ONTHOUDEN

Is een array niet gedefinieerd door een DIM en komt de computer toch zo'n array tegen, dan neemt hij automatisch aan dat deze array bestaat uit tien elementen. Echter, ongedefinieerde arrays kunnen niet meer dan tien elementen bevatten, zodat er een 'Subscript wrong'-foutmelding optreedt. U kunt hieruit concluderen dat, als u een array nodig heeft met minder dan tien elementen, u geen enkelvoudig dimensionale array hoeft te definiëren. Dus DIM A%(5) is niet nodig!

De maximale dimensie van een array is 255 elementen.

De minimale waarde van een index is nul. Daarom begint de array vanaf het nulste element.

DIM A(5) levert op A(0), A(1), A(2), A(3), A(4) en A(5); dus zes elementen.

Indien een array initieel wordt gedefinieerd door een DIM-opdracht, worden alle elementen op nul gezet.

FOUTEN

Verwijst u een array buiten zijn limieten dan krijgt u de foutmelding 'Subscript out of range'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Sectie 1, Hoofdstuk 12: Het lezen van gegevens in arrays

Sectie 1, Hoofdstuk 13: Gegevensmanipulatie en sorteren

Sectie 2, Hoofdstuk 2: Constanten en variabelen

DRAW

BESCHRIJVING

DRAW stelt u in staat om plaatjes te tekenen volgens de grafische macrotaal (GML). De grafische macrotaal is geschreven in de vorm van strings en DRAW vertelt de computer volgens deze subtaal te tekenen.
U moet onthouden dat er een grafische cursor is die overal op het scherm kan worden geplaatst. De grafische macrotaal bestuurt de bewegingen en de activiteiten van deze cursor volgens de door u geprogrammeerde GML.

SYNTAX

DRAW "<string>"
DRAW <string-var>
DRAW <string-uitdrukking>

Grafische macrotaal-commando's:
Omhoog, omlaag, links en rechts.

U<n>	: tekent omhoog
D<n>	: tekent omlaag
L<n>	: tekent naar links
R<n>	: tekent naar rechts

<n> geeft hier de verplaatsing aan. Deze hangt af van de gekozen schaal S (zie verderop in deze beschrijving).

Diagonaal tekenen

E<n>	: tekent diagonaal rechts omhoog
F<n>	: tekent diagonaal rechts omlaag
G<n>	: tekent diagonaal links omlaag
H<n>	: tekent diagonaal links omhoog

<n> wijkt hier iets af van het horizontaal en verticaal tekenen. E<n> bijvoorbeeld, tekent <n> pixels omhoog en <n> pixels naar rechts, dus niet <n> pixels diagonaal.

Tekenen naar specifieke coördinaten

Om naar specifieke coördinaten op het scherm te tekenen, kunt u het M-commando gebruiken dat de grafische cursor verplaatst van de laatste positie naar de x- en y-coördinaten gedurende het tekenen.

M<x>,<y>	: tekent naar de coördinaten x,y

U kunt ook relatief tekenen ten opzichte van het laatste punt, met behulp van de '+'- of '-'-voorvoegsels bij de x- en y-coördinaten

M+<x>,<y>	: M-<x>,<y>
M+<x>,-<y>	: M-<x>,-<y>

Verplaatsing van de cursor zonder te tekenen
Er zijn momenten dat u de cursor alleen wilt verplaatsen zonder daarbij te tekenen. Gebruik hiervoor het B-voorvoegsel; B staat voor Blank. Dit B-voorvoegsel stopt het tekenen en verplaatst de cursor.
Voorvoegsel B voor alle bovenstaande tekencommando's:

B : verplaatst zonder tekenen

De volgende combinaties zijn mogelijk:

BU<n>	: verplaatst omhoog
BD<n>	: verplaatst omlaag
BL<n>	: verplaatst naar links
BR<n>	: verplaatst naar rechts
BE<n>	: verplaatst rechts omhoog
BF<n>	: verplaatst rechts omlaag
BG<n>	: verplaatst links omlaag
BH<n>	: verplaatst links omhoog
BM<x>,-<y>	: verplaatst absoluut naar coördinaten x,y
BM+<x>,-<y>	: BM-<x>,<y>; verplaatst relatief
BM+<x>,-<y>	: BM-<x>,-<y>; met stap x en y

Terugkeren naar de beginpositie na het tekenen van een lijn
Wilt u terugkeren naar het punt waar u bent begonnen met het tekenen van een lijn, dan kunt u het voorvoegsel N gebruiken. Dit voorvoegsel brengt u terug naar het punt waar u bent begonnen.

Voorvoegsel N : tekent wel, maar plaatst de cursor terug op zijn beginpositie.

De volgende combinaties zijn mogelijk:

NU<n>	: tekent omhoog, keert dan terug
ND<n>	: tekent omlaag, keert dan terug
NL<n>	: tekent links, keert dan terug
NR<n>	: tekent rechts, keert dan terug
NE<n>	: tekent rechts omhoog, keert dan terug
NF<n>	: tekent rechts omlaag, keert dan terug
NG<n>	: tekent links omlaag, keert dan terug
NH<n>	: tekent links omhoog, keert dan terug
NM<x>,<y>	: tekent absoluut naar het punt met de coördinaten x,y en keert dan terug
NM+<x>,<y>	: NM-<x>,<y>; verplaatst relatief met stap x en y
NM+<x>,-<y>	: NM<x>,-<y>; keert dan terug

Roteren van de richting van een relatieve verplaatsing
Met behulp van het hoekcommando A kunt u de as van de relatieve verplaatsingscommando's, zoals U, D, L, R, F, E, G en H laten roteren.

Hoekcommando A

A<n> : <n> kan zijn 0, 1, 2 of 3
A0 : met 0 graden
A1 : roteert tegen de klok in, 90 graden
A2 : roteert tegen de klok in, 180 graden
A3 : roteert tegen de klok in, 270 graden

Het zetten van de kleur met GML

U kunt het C-commando gebruiken om de kleur te wijzigen die binnen de GML wordt toegepast. Binnen één enkel DRAW-commando kunt u de tekenkleur zo vaak wijzigen als u wilt.

Kleurcommando C-voorvoegsel

C<n> : zet de kleur tijdens het tekenen
<n> : is een integer tussen 0 en 15

C0	transparant	C8	middelrood
C1	zwart	C9	lichtrood
C2	middelgroen	C10	donkergeel
C3	lichtgroen	C11	lichtgeel
C4	donkerblauw	C12	donkergroen
C5	lichtblauw	C13	paarsrood
C6	donkerrood	C14	grijs
C7	cyaanblauw	C15	wit

Wijzigen van de tekenschaal

Schaalcommando S-voorvoegsel:

S<n> : hierin kan <n> liggen tussen 0 en 255
<schaalfactor>=<n>/4

Ten gevolge van dit commando tekent S1 éénvierde van de lengte zoals opgegeven in U, D, L, R enz.
S0 en S4 zijn identiek en resulteren in geen schaal.
S8 zet de schaal op dubbele grootte van de vervangingsschaal (S4).

Het gebruik van substrings

X<string-var>; geeft aan dat wat zich in de stringvariabele bevindt, dient te worden uitgevoerd. In de GML is het mogelijk een string te hebben die de tekencommando's bevat. De puntkomma is altijd noodzakelijk.

```
DRAW "X<string-var>"
```

Het gebruik van numerieke variabelen binnen de GML

In alle grafische macrocommando's kunnen <n>, <x> en <y> variabelen zijn.

Zij moeten echter dan het voorvoegsel '=' hebben en vereisen een puntkomma aan het einde, dus:

```
<num-var>;        =<n>;
                  =<x>;
                  =<y>;
```

Voorbeeld van een numerieke variabele G%:

```
DRAW "U=G%;"
```

VOORBEELDEN

```
10 SCREEN 2
20 A$="R50U50L50D50"
30 DRAW "BM100,150"
40 DRAW "S0XA$;"
50 IF INKEY$="" THEN 50
60 FOR I=1 TO 10
70 DRAW "S=I;XA$;"
80 NEXT I
90 GOTO 90
```

REGEL 10 selecteert HIRES-scherm.
REGEL 20 A$ bevat de GML.
REGEL 30 plaatst de grafische cursor op 100,150.
REGEL 40 tekent een kubus A$, normale grootte.
REGEL 50 wacht op het indrukken van een toets.
REGEL 70 tekent een kubus A$ met grootte 1.
REGEL 90 bevriest het grafische scherm.
OPMERKING: A$ leest '50 pixels naar rechts, 50 pixels omhoog, 50 pixels naar links, 50 pixels omlaag'.

PUNTEN OM TE ONTHOUDEN

Het X-commando is zeer handig omdat u een gedeelte van een tekening kunt definiëren en dit overal op kunt nemen. Het idee heeft meer iets van een grafische subroutine binnen een grafisch commando.

Het X-commando biedt de mogelijkheid een tekening op te nemen van meer dan 255 tekens, indien expliciet geplaatst in een DRAW-opdracht.

S-commando: Wijzigt u S4 dan onthoudt de computer de nieuwe waarde en tekent alles hierna met deze schaal. Wilt u terug naar de oorspronkelijke schaal dan dient u S4 in een DRAW-commando op te nemen.

BM verplaatst de grafische cursor naar een specifieke plaats op het scherm zonder iets te tekenen. Dit is de aangewezen manier om een object te positioneren. Er zijn echter vele andere manieren om dit te bereiken. Is

het oorspronkelijke punt niet opgegeven door BM, dan begint de computer vanaf het laatst bezochte punt, dus daar waar de grafische cursor na de laatste DRAW-opdracht is gestopt.
U kunt het oorspronkelijke punt bepalen met PSET of met het M-commando.
Zodra een hoekcommando A is uitgevoerd zullen alle latere DRAW-commando's worden geroteerd, tenzij u deze terugzet in de oorspronkelijke richting met een ander A-commando. De hoek wordt niet op een vervangingswaarde gezet na het uitvoeren van een DRAW-commando of na het beëindigen van een programma.
De huidige voorgrondkleur blijft ongewijzigd na het uitvoeren van het DRAW-commando dat het C-commando bevat.

FOUTEN

U kunt DRAW alleen in de grafische modus gebruiken. Past u een verkeerde GML-syntax toe dan treedt een 'Illegal Function Call'-foutmelding op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

COLOR
SCREEN
Sectie 1, Hoofdstuk 26: De grafische macrotaal

ELSE

BESCHRIJVING
ELSE is een onderdeel van de IF...THEN...ELSE-structuur. ELSE laat de computer weten dat indien niet aan de <conditie> van IF kan worden voldaan, de opdrachten na THEN moeten worden overgeslagen; de opdrachten die volgen op ELSE moeten hiervoor in de plaats worden uitgevoerd.
ELSE GOTO <regel> kan worden afgekort tot ELSE <regel>. Meerdere IF...THEN ...ELSE-instructies kunnen worden opgezet, ook nesten hiervan is mogelijk. De enige beperking hier is het aantal tekens in één regel: maximaal 255.

SYNTAX
IF <conditie> THEN <opdrachten> ELSE <opdrachten>
IF <conditie> THEN <opdrachten> ELSE <regel>

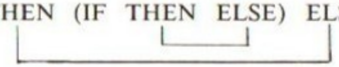
VOORBEELDEN
Eenvoudige IF...THEN...ELSE-structuur:

```
10 INPUT D$
20 IF D$="NOORD" THEN PRINT "U KOMT EEN
VRESELIJK MONSTER TEGEN!" ELSE PRINT "U
GAAT NAAR ";D$
30 GOTO 10
```

```
RUN
? NOORD
U KOMT EEN VRESELIJK MONSTER TEGEN!
? ZUID
U GAAT NAAR ZUID
```

PUNTEN OM TE ONTHOUDEN
Staan er minder ELSE-instructies dan THEN-instructies in een geneste IF... THEN...ELSE-structuur dan wordt iedere ELSE gerelateerd aan de dichtstbij gelegen THEN.
Voorbeeld:

IF THEN (IF THEN (IF THEN ELSE) ELSE)



IF...THEN...ELSE-opdrachten neigen al gauw veel te lang te worden om in een enkele regel te passen, omdat u meerdere opdrachten kunt opgeven na THEN en ELSE. Is dit het geval dan doet u er goed aan subroutines te definiëren met behulp van de GOSUB-opdracht. IF...THEN...ELSE kan leiden tot een spaghetti-programma als u niet uitkijkt.

FOUTEN
Het blijkt dat er vaak fouten worden gemaakt in het <conditie>-gedeelte van IF...THEN...ELSE-instructies.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
AND
OR
NOT
IF
THEN
Sectie 1, Hoofdstuk 6: Het gebruik van condities
Sectie 2, Hoofdstuk 7: BOOLEAN II: De IF...THEN...ELSE-instructie

END

BESCHRIJVING
END geeft de computer aan dat het einde van het programma is bereikt. Hierna gaat de computer over in de commandomode.
U mag END in ieder gedeelte van uw programma opnemen, net zo vaak als u wenst.
U zult wellicht een END plaatsen voor een subroutine, om zo te voorkomen dat uw programma hier per ongeluk in terecht komt.
END is niet noodzakelijk aan het einde van uw programma, omdat de computer veronderstelt dat het einde is bereikt zodra blijkt dat er geen instructies meer volgen.
Zodra de computer END tegenkomt sluit hij alle geopende bestanden af. Dit verschilt dus ten opzichte van STOP, die weliswaar het programma laat stoppen, maar geen bestand afsluit.

SYNTAX
END

VOORBEELDEN
```
9999 END
```

PUNTEN OM TE ONTHOUDEN
Het verschil tussen STOP en END is dat STOP een 'Break'-boodschap stuurt maar END niet. STOP sluit ook geen geopende bestanden af.

FOUTEN
Heel duidelijk: indien u END plaatst in uw programma op een plaats waar u uw programma niet wilt laten stoppen, volgt een vroegtijdig einde van uw programma. Juist!

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
STOP
Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's.

EOF Einde van een bestand

BESCHRIJVING
EOF laat u weten dat het einde van een bestand is bereikt. EOF geeft -1 (waar) als het einde is bereikt, 0 (niet waar) als dit niet zo is.
EOF is een vitaal onderdeel bij bestandsmanipulatie en wordt gebruikt met OPEN enz.

SYNTAX
EOF (<bestandsnummer>)

VOORBEELDEN
```
IF EOF(4)=-1 THEN PRINT "EINDE BESTAND":END
```

FOUTEN
Heeft u het einde van een bestand bereikt en heeft u geen EOF gebruikt, dan volgt een 'Input Past end'-foutboodschap.
Verwijst u naar een niet-geopend bestand in <bestandsnummer> dan treedt 'File not open' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
OPEN
CLOSE
INPUT#
Sectie 2, Hoofdstuk 19: Hoe bestanden worden gebruikt

EQV Equivalent

BESCHRIJVING
EQV is één van de BOOLEAN logische bewerkers die wordt gebruikt in binaire manipulatie. De werking ervan wordt weergegeven in de volgende waarheidstabel:

EQV	X	Y	X AND Y
	0	0	1
	1	0	0
	0	1	0
	1	1	1

Voorbeeld: 51 EQV 74

	51	0000000000110011
EQV	74	0000000001001010
	-122	1111111110000110

SYNTAX
<numeriek>EQV<numeriek>
<numeriek> moet liggen tussen -32768 en 32767. Breuken van decimale getallen worden afgerond.

VOORBEELDEN

```
10 A=51
20 B=74
30 PRINT A;" EQV ";B
40 PRINT A EQV B,BIN$(A EQV B)
```

```
RUN
51 EQV 74
-122       1111111110000110
```

PUNTEN OM TE ONTHOUDEN
De volgende relaties zijn waar:

X EQV X = -1
-X EQV X = 1

FOUTEN
Er treedt een 'Overflow error' op indien het argument buiten de toegestane marge ligt (-32768 tot 32767).

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

AND
OR
XOR
IMP
NOT
Sectie 2, Hoofdstuk 6: BOOLEAN algebra

ERASE Verwijder een array

BESCHRIJVING
Deze opdracht stelt u in staat om door DIM gecreëerde arrays te verwijderen. Dit biedt de mogelijkheid om een nieuwe dimensie toe te kennen aan de verwijderde array.

SYNTAX
ERASE <naam van array>
ERASE <naam van array>,<naam van array>, ...

VOORBEELDEN

```
10 A%=99
20 DIM A%(150)
30 FOR I=1 TO 150
40 A%(I)=I
50 NEXT I
60 ERASE A%
70 DIM A%(1000)
80 FOR I=1 TO 1000
90 A%(I)=I
100 NEXT I
110 PRINT A%
```

```
RUN
 99
```

PUNTEN OM TE ONTHOUDEN
Een variabele met dezelfde naam als die van een array wordt niet beïnvloed door het commando ERASE, zoals bovenstaand voorbeeld laat zien.
Alleen de naam van de te verwijderen array is nodig in de ERASE-opdracht. Dit betekent dat u niet hoeft te schrijven ERASE A%(150); ERASE A% is voldoende.

FOUTEN
Wordt een niet-bestaande array verwijderd met ERASE dan treedt een 'Illegal Function Call'-foutmelding op.
Probeert u een nieuwe dimensie toe te kennen aan een array zonder eerst ERASE uit te voeren, dan treedt een 'Redimensionised array'-foutmelding op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
DIM
Sectie 1, Hoofdstuk 12: Het lezen van gegevens in arrays

ERL Foutregel

BESCHRIJVING
ERL is een gereserveerde variabele die het regelnummer bevat van de regel waarin een fout werd geconstateerd. Deze variabele wordt gebruikt in foutdetectieroutines.

SYNTAX
<num var>=ERL
PRINT ERL

VOORBEELDEN

```
10 ON ERROR GOTO 1000
20 PRINT "Z80"
30 PRINT "TMS 9918"
40 PRONT "FOUT!!"
50 END
1000 E%=ERL
1010 PRINT "Een domme fout in regel ";E%
1020 END
```

```
RUN
Z80
TMS 9918
Een domme fout in regel 40
Ok
```

PUNTEN OM TE ONTHOUDEN
Treedt er een fout op in de directe modus dan bevat ERL 65535. ERL is een gereserveerde variabele; u kunt er dus geen getal aan toekennen.

FOUTEN
Gebruik ERL in foutdetectieroutines om de exacte plaats van de fout te bepalen.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ON ERROR GOTO
ERROR
ERR
Sectie 2, Hoofdstuk 2: Foutafhandeling

ERR Foutcodenummer

BESCHRIJVING
Dit is een gereserveerde variabele waarin de computer de foutcode bewaart nadat er een fout is opgetreden in een programma. Deze variabele wordt hoofdzakelijk gebruikt in foutdetectieroutines.
De waarde van ERR kan variëren tussen 1 en 255. ERR kan ook worden gebruikt ten behoeve van door de gebruiker zelf gedefinieerde foutboodschappen. Een aantal nummers in de range van 1 tot 255 is vrij voor gebruik.
ERR wordt gewoonlijk gebruikt bij het creëren van nieuwe foutmeldingen met behulp van ERROR (zie onderstaand voorbeeld).

SYNTAX
<num var>=ERR

VOORBEELDEN
In deze routine worden alle commando's die DOODT bevatten behandeld als een nieuwe fout (nr. 255) in de foutdetectieroutine met behulp van ERR.

```
10 ON ERROR GOTO 1000
20 INPUT "MAJESTEIT, WAT IS UW OPDRACHT";A$
30 IF INSTR(A$,"DOODT") THEN ERROR 255
40 PRINT "DOEN WE.";END
...
...
1000 IF ERR=255 THEN PRINT "DODEN IS NIET
TOEGESTAAN: EINDE VAN DIT PROGRAMMA":END
1010 END
```

```
RUN
MAJESTEIT, WAT IS UW OPDRACHT? DOODT HEM
DODEN IS NIET TOEGESTAAN: EINDE VAN DIT PROGRAMMA
OK
```

PUNTEN OM TE ONTHOUDEN
ERR is een gereserveerde variabele: u kunt er daarom geen waarde aan toekennen.

FOUTEN
Gebruik deze functie om fouten in uw programma op te sporen.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ERL, ERROR, ON ERROR, RESUME
Sectie 2, Hoofdstuk 10: Foutafhandeling

ERROR

BESCHRIJVING
Er bestaan in principe twee manieren om ERROR toe te passen:
1. Bij de simulatie van het optreden van een fout.
2. Bij het creëren van door de gebruiker zelf gedefinieerde fouten met behulp van ON ERROR GOTO.

1. Een ERROR-opdracht met een integer-argument misleidt de computer, die denkt met een fout te maken te hebben. De computer zal het programma beëindigen en de corresponderende foutboodschap afdrukken samen met het regelnummer waarin de fout is opgetreden.
2. Een ERROR-opdracht samen met het gebruik van de ON ERROR GOTO fout-detectieopdracht, stelt u in staat uw eigen fouten te creëren.

Er bestaan ongeveer 36 fouten, waarvan de foutcode ligt tussen 1 en 60, althans in de huidige versie van MSX-BASIC. De codenummers 61 tot 255 kunnen worden gebruikt door de programmeur.
Om een eigen fout te creëren dient u uw programma eerst in de foutdetectie-modus te plaatsen door het uitvoeren van een ON ERROR GOTO <regel>-opdracht aan het begin van uw programma. Als nu ergens midden in uw programma een fout optreedt waarvoor u de foutroutine wilt aanroepen, dan kunt u dit bereiken met:

IF <conditie> THEN ERROR <foutcode>

ON ERROR GOTO wordt geactiveerd en de computer zal meteen beginnen met de uitvoering van de foutdetectiesubroutine. Binnen deze subroutine moet u één regel hebben:

IF ERR=<foutcode> THEN PRINT "Foutboodschap"

De foutdetectieroutine kan zowel stoppen met de uitvoering van het programma via END, als continueren vanaf een opgegeven regel met RESUME.

SYNTAX
ERROR <foutcode>
waarin <foutcode>=0 tot 255

VOORBEELDEN
1. Een kort programma dat een fout simuleert.

```
10 ERROR 11
```

```
RUN
Delen door nul in 10
```

2. In deze routine worden alle commando's die DOODT bevatten behandeld als een nieuwe fout (nr. 255) in de foutdetectieroutine, met behulp van de ERR-functie.

```
10 ON ERROR GOTO 1000
20 INPUT "KONING, WAT IS UW OPDRACHT";A$
30 IF INSTR(A$,"DOODT") THEN ERROR 255
40 PRINT "DOEN WE.":END
...
...
1000 IF ERROR=255 THEN PRINT "DODEN IS NIET
TOEGESTAAN IN DIT AVONTUUR.":RESUME 20
1010 END
```

```
RUN
KONING WAT IS UW OPDRACHT? DOODT HEM
DODEN IS NIET TOEGESTAAN IN DIT AVONTUUR.
KONING, WAT IS UW OPDRACHT? GA OOSTWAARTS
DOEN WE.
```

PUNTEN OM TE ONTHOUDEN

Het is gebruikelijk in MSX-BASIC dat bij het zelf definiëren van fouten omlaag wordt gewerkt vanaf 255, opdat u problemen kunt vermijden met toekomstige ontwikkelingen van foutboodschappen, gemaakt door de fabrikanten van de MSX-computers.

Heeft een foutcode geen eerder genoemde foutboodschap en komt de computer deze tegen, dan drukt het programma 'Unprintable Error' af en stopt.

Er wordt geen foutdetectie uitgevoerd tijdens de uitvoering van een foutdetectieroutine. Blijkt er een fout te zitten in de foutdetectieroutine dan wordt de foutboodschap gegeven en het programma gestopt.

Zodra de foutdetectie wordt geactiveerd, constateert de computer ook fouten in de directe modus.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

ON ERROR GOTO
ERL
ERR
RESUME
Sectie 2, Hoofdstuk 10: Foutafhandeling

EXP Exponent

BESCHRIJVING
Deze wiskundige functie geeft de exponentiële waarde van een gegeven argument X.

```
e^x
```

SYNTAX
EXP(<num const>)
EXP(<num var>)
EXP(<num uitdrukking>)

VOORBEELDEN
```
PRINT EXP(1)
 2.7182818284588
```

PUNTEN OM TE ONTHOUDEN
Deze functie berekent met dubbele nauwkeurigheid tot op 14 decimalen.

FOUTEN
Is de exponent te groot voor de computer om te behandelen, dan treedt een 'Overflow error' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
LOG
Sectie 1, Hoofdstuk 19: Wiskundige functies

FIX Geef integer deel

BESCHRIJVING
Deze functie geeft het integere gedeelte terug van een argument. De breuken worden afgerond.
FIX is equivalent aan SGN(x)*INT(ABS(x)).
FIX geeft de dichtstbij gelegen lagere integer voor positieve argumenten en de dichtstbij gelegen hogere integer voor negatieve argumenten.

FIX(1.87)=1
FIX(-12.88)=-12

SYNTAX
FIX(<num const>)
FIX(<num var>)
FIX(<num uitdrukking>)

VOORBEELDEN
```
PRINT FIX(-1.2209)
```
resulteert in -1
en
```
A%=FIX(10.99)
```
maakt A%=10

PUNTEN OM TE ONTHOUDEN
De INT-functie geeft de dichtstbij gelegen lagere waarde bij zowel negatieve als positieve argumenten. Bijvoorbeeld: INT(-1.3)=-2. Er is dus weinig verschil tussen INT en FIX.

FOUTEN
Het betreft hier een numerieke functie, dus vergis u niet met de strings, omdat anders een 'Type Mismatch'-fout optreedt.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
INT
Sectie 1, Hoofdstuk 15: Functies

FOR FOR...TO...NEXT-lus

BESCHRIJVING

FOR is één van de instructies om lussen te creëren. De reeks voor de variabeleteller is gedefinieerd. De waarde waarmee de teller wordt verhoogd na iedere doorloop van een lus kan worden opgegeven met STEP.
Bijvoorbeeld:

```
10 FOR I=1 TO 3
20 PRINT I
30 NEXT I
```

geeft de getallen 1, 2 en 3.
De teller neemt initieel de waarde 1 aan en voert de opdrachten tussen FOR en NEXT uit, dit is regel 20 PRINT I. Als regel 30 wordt bereikt, NEXT I, springt de computer terug naar regel 10, verhoogt de teller met 1 en herhaalt de lus: dit gaat zo door totdat de grenswaarde 3 is bereikt.
Tot nu toe hebben we alleen het FOR...TO...NEXT-formaat bekeken, maar we kunnen ook STEP introduceren in deze regel om aan te geven met welke waarde de teller dient te worden aangepast na elke doorloop van een lus. We passen bovenstaand programma als volgt aan:

```
10 FOR I=1 TO 3 STEP 0.5
20 PRINT I
30 NEXT I
```

geeft 1, 1.5, 2, 2.5 en 3.
Ook is een geneste lus mogelijk, dit is een lus in een lus, zoals hieronder:

```
10 FOR I=1 TO 5
20 FOR J=1 TO 5
30 PRINT I,J
40 NEXT J
50 NEXT I
```

welke achtereenvolgens afdrukt:

```
1        1
1        2
1        3
enz.
```

Er zijn een aantal regels voor geneste lussen:

1. Iedere lus moet een andere variabele hebben als teller.
2. De NEXT van de binnenste lus dient eerder te worden uitgevoerd dan die van een buitengelegen lus.
3. U kunt zoveel nesten als u wilt. De enige beperking is hier de geheugencapaciteit.

4. Een enkele NEXT-opdracht, gevolgd door een reeks tellers kan een hele serie NEXT-opdrachten vervangen.

SYNTAX

FOR <num var>=<numeriek>TO<numeriek>
FOR <num var>=<numeriek>TO<numeriek>STEP<numeriek>

VOORBEELDEN

U kunt een negatieve verhoging gebruiken, zoals hieronder blijkt:

```
10 FOR F=10 TO 5 STEP -1
20 PRINT F
30 NEXT F
```

```
RUN
10
9
8
7
6
5
```

PUNTEN OM TE ONTHOUDEN

Als u midden in een lus uit de lus springt dient u altijd terug te keren in de lus. Daartoe kunt u GOSUB gebruiken, welke u in de lus doet terugkeren na RETURN. Uit de lus springen met GOTO is slecht programmeren en dient te worden vermeden.

FOUTEN

Er moet een corresponderende NEXT bestaan voor iedere FOR-opdracht, anders treedt een 'NEXT without FOR'-foutmelding op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

NEXT
STEP
TO
Sectie 1, Hoofdstuk 5: Het gebruik van lussen

FRE Hoeveelheid vrije geheugenruimte

BESCHRIJVING
Deze systeemvariabele vertelt hoeveel geheugenruimte u nog over heeft in het BASIC-programmagebied en stringopslaggebied.

SYNTAX
FRE(0) geeft de vrije geheugenruimte in het programmagebied.
FRE("") geeft de vrije geheugenruimte in het stringgebied.

VOORBEELDEN
```
PRINT FRE(0)
 28815
PRINT FRE("")
 200
```

PUNTEN OM TE ONTHOUDEN
FRE(0) is equivalent aan het geheugengebied aangegeven door FREE in de geheugen-lay-out.

FOUTEN
U heeft een dummy-argument (0) en ("") nodig in de FRE-functie.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 17: Nuttige commando's en tips voor het schrijven van programma's
Sectie 2, Hoofdstuk 20: Geheugenorganisatie

GOSUB Ga naar een subroutine

BESCHRIJVING
In BASIC-programmering blijkt het vaak het geval te zijn dat u eenzelfde routine op verschillende plaatsen nodig heeft. U kunt een subroutine plaatsen op een andere locatie dan waar hij nodig is en u kunt GOSUB gebruiken om de routine vanaf het aangegeven regelnummer aan te roepen, zo vaak u maar wilt.
Zodra BASIC een GOSUB tegenkomt wordt direct naar het aangegeven regelnummer gesprongen en wordt de uitvoering vanaf dat punt voortgezet. Als de computer RETURN heeft bereikt wordt teruggekeerd naar het hoofdprogramma van waaruit werd gesprongen.
U kunt zelfs een subroutine binnen een subroutine aanroepen. In feite kunt u net zoveel subroutines nesten als u wilt, zolang hiervoor voldoende geheugenruimte bestaat. Het is mogelijk om een recursieve subroutine te hebben die zichzelf aanroept.

SYNTAX
GOSUB <regel>

VOORBEELDEN
Een voorbeeld van een recursieve GOSUB. De subroutine roept zichzelf aan bij regel 110 totdat aan de conditie wordt voldaan.

```
10 PRINT "BEGIN"
20 A=1
30 GOSUB 100
40 PRINT A
50 END
90 REM SUBROUTINE
100 A=A+1
110 IF A<>100 THEN GOSUB 100
120 RETURN
```

```
RUN
BEGIN
100
```

PUNTEN OM TE ONTHOUDEN
U dient te voorkomen dat u naar een regel springt met een REM-opdracht, omdat het kan gebeuren dat vanwege geheugenruimtegebrek de REM-regels worden verwijderd.
Het is niet mogelijk om een variabele uitdrukking te gebruiken voor <regelnummer>, dus GOSUB A%*10 is fout.

Het is een gewoonte bij computerprogrammering de subroutines gescheiden te houden van het hoofdprogramma; voor de leesbaarheid doet u er goed aan iedere subroutine te voorzien van een REM-regel.
Meerdere RETURN-opdrachten kunnen worden toegepast in een programma voor één en dezelfde subroutine.
GOSUB wordt veel toegepast bij evenement-interruptie-opdrachten zoals bij ON SPRITE enz. Zie voor meer details de hiertoe relevante gedeelten in dit boek.

FOUTEN
Wordt vanuit GOSUB naar een niet-bestaand regelnummer verwezen, dan treedt een 'Undefined Line Number'-foutmelding op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
RETURN
ON...GOSUB
ON INTERVAL GOSUB
ON KEY...GASUB
ON SPRITE GOSUB
ON STOP GOSUB
ON STRIG GOSUB
Sectie 1, Hoofdstuk 17: Gestructureerd programmeren

GOTO Spring naar een regel

BESCHRIJVING

GOTO laat de computer weten dat naar een specifiek regelnummer dient te worden gesprongen en vanaf die regel moet de uitvoering van het programma worden voortgezet.
In een IF...THEN...GOTO-opdracht kunt u THEN GOTO vervangen door alleen GOTO of THEN. Uw computer begrijpt wat u bedoelt.

SYNTAX

GOTO <regel>

VOORBEELDEN

```
10 PRINT "WELKOM BIJ UW MSX"
20 GOTO 10
```

```
RUN
WELKOM BIJ UW MSX
WELKOM BIJ UW MSX
WELKOM BIJ UW MSX
Break in 10
```

Het afdrukken blijft doorgaan totdat u <CTRL> <STOP> indrukt.

PUNTEN OM TE ONTHOUDEN

U kunt een GOTO naar de regel van de GOTO zelf laten uitvoeren. Hiermee kunt u een bepaalde controle herhaaldelijk uitvoeren of een grafisch scherm bevriezen, bijvoorbeeld:

```
100 IF INKEY$<>"A" GOTO 100
```

FOUTEN

Verwijst GOTO naar een niet-bestaand regelnummer dan volgt een 'Undefined Line Number'-foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

ON GOTO
ON ERROR GOTO
Sectie 1, Hoofdstuk 3: Het schrijven van een programma

HEX$ HEX string

BESCHRIJVING
HEX$ geeft de hexadecimale waarde van een decimaal integer getal in een string. Bijvoorbeeld, HEX$(255) geeft de string FF.

SYNTAX
HEX$(<integer numeriek>)
Er kan gebruik worden gemaakt van waarden die liggen tussen -32768 en 65535.
Voor negatieve integere waarden wordt de tweecomplementvorm gebruikt, dus HEX$(65535-x)=HEX$(-x).

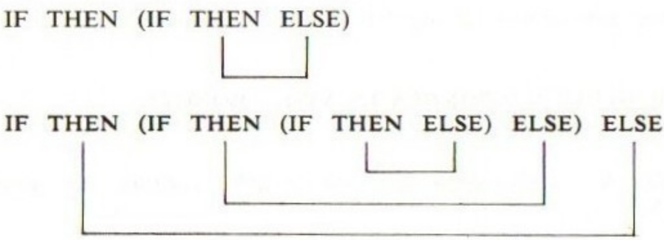
PUNTEN OM TE ONTHOUDEN
&H<getal> geeft de decimale waarde van een hex-getal.

FOUTEN
Het argument moet integer zijn, terwijl het resultaat een string is.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
OCT$
BIN$
Sectie 2, Hoofdstuk 5: Overzicht van de getallensystemen die worden gebruikt bij MSX

IF

BESCHRIJVING
De IF-opdracht controleert een bepaalde conditie of een combinatie van condities en reageert hier vervolgens op.
De conditietestopdrachten die betrekking hebben op IF hebben de volgende formaten:
-IF- indien conditie is waar -THEN- voer dan alles uit wat volgt op THEN.
-IF- indien conditie niet-waar -THEN- negeer dan alles achter THEN, maar zet de uitvoering voort vanaf de volgende regel.
ELSE is een onderdeel van de IF...THEN...ELSE-structuur. ELSE vertelt de computer dat, indien aan een conditie niet kan worden voldaan (conditie is niet waar), de opdrachten die volgen op THEN moeten worden genegeerd en alles wat volgt op ELSE hiervoor in de plaats dient te worden uitgevoerd.
Meerdere IF...THEN...ELSE-opdrachten kunnen worden opgezet, evenals het nesten van IF...THEN...ELSE-opdrachten. Zij worden alleen beperkt door de maximale lengte van een regel, dit is 255 tekens. Zijn er minder ELSE-woorden dan THEN's, dan wordt iedere ELSE gerelateerd aan de dichtstbij gelegen THEN.



De IF...THEN...GOTO <regel>-structuur kan als volgt worden afgekort:

1. IF...THEN <regel>
2. IF...GOTO <regel>

ELSE GOTO <regel> kan worden afgekort tot ELSE <regel>.

SYNTAX
IF <conditie> THEN <opdrachten>
IF <conditie> THEN <regel>
GOTO
IF <conditie> THEN <opdracht/regel> ELSE <opdracht/regel>
GOTO GOTO

VOORBEELDEN
Regel 50 is een eenvoudige IF...THEN-structuur.
Regel 60 is een IF...THEN...ELSE...IF...THEN...ELSE-structuur.

```
10 PRINT "GEEF DRIE VERSCHILLENDE GETALLEN OP"
20 INPUT "A";A%
30 INPUT "B";B%
40 INPUT "C";C%
50 IF A%=B% OR A%=C% OR B%=C% THEN 10
60 IF A%<B% AND A%<C% THEN PRINT "A" ELSE
IF B%<A% AND B%<C% THEN PRINT "B" ELSE
PRINT "C"
90 PRINT " IS HET KLEINSTE GETAL"
```

```
RUN
GEEF DRIE VERSCHILLENDE GETALLEN OP
A? 8
B? 4
C? 3
C
IS HET KLEINSTE GETAL
```

PUNTEN OM TE ONTHOUDEN

IF...THEN...ELSE-opdrachten neigen al snel te lang te worden om nog in een enkele regel te passen, omdat meerdere opdrachten mogelijk zijn na iedere THEN en ELSE. Blijkt dit het geval te zijn, dan doet u er goed aan gebruik te maken van de GOSUB-opdracht.

Het wordt aangeraden om geneste IF...THEN...ELSE-opdrachten te vermijden omdat deze al snel leiden tot verwarring. U heeft al gauw een spaghetti-programma als u niet oppast.

FOUTEN

Veelal blijken de fouten te worden gemaakt in het <conditie>-gedeelte van de IF...THEN...ELSE-opdrachten.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

THEN
ELSE
AND
OR
NOT
Sectie 1, Hoofdstuk 6: Het gebruik van condities
Sectie 2, Hoofdstuk 7: BOOLEAN II: de IF...THEN...ELSE

IMP Implicatie

BESCHRIJVING
Dit is één van de BOOLEAN logische bewerkers. De BOOLEAN algebraïsche bewerking wordt voor IMP in de volgende waarheidstabel weergegeven:

IMP	X	Y	X IMP Y
	0	0	1
	1	0	0
	0	1	1
	1	1	1

100 IMP 50 wordt als volgt berekend:

	X	100	0000000001100100
IMP	Y	50	0000000000110010
		-69	1111111110111011

SYNTAX
<num var>=<num const> IMP <num const>
<num var=<num var> IMP <num uitdrukking>
en andere numerieke combinaties.

VOORBEELDEN

```
10 S=99
20 PRINT S IMP S
30 PRINT (-S IMP S)
40 PRINT (S IMP -S)
```

```
RUN
-1
99
-99
```

PUNTEN OM TE ONTHOUDEN
De volgende relaties zijn waar:

X IMP X = -1
-X IMP X = X
X IMP -X = -X

IMP is een 16-bit bewerker.

FOUTEN
Er treedt een 'Overflow error' op indien één van de argumenten buiten de toegestane marge valt.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
AND
OR
NOT
XOR
EQV
Sectie 2, Hoofdstuk 6: BOOLEAN algebra

INKEY$

BESCHRIJVING
Deze functie tast het toetsenbord af en controleert daarbij of er een toets wordt ingedrukt; is dit zo, dan geeft INKEY$ het bijbehorende teken. Wordt er geen toets ingedrukt dan geeft INKEY$ een nulstring.

SYNTAX
<string var>=INKEY$

VOORBEELDEN
Dit voorbeeld laat zien hoe u de computer kunt laten wachten totdat er een toets wordt ingedrukt. Regel 20 controleert het indrukken van de toets en slaat het eventueel ingedrukte teken op in A$. Wordt er niets ingedrukt dan wordt teruggekeerd naar regel 20 om opnieuw te controleren of er een toets wordt ingedrukt.

```
10 PRINT "DRUK OP EEN TOETS OM HET PROGRAMMA
TE BEEINDIGEN"
20 A$=INKEY$:IF A$="" THEN 20 ELSE STOP
```

```
RUN
DRUK OP EEN TOETS OM HET PROGRAMMA TE
BEEINDIGEN
Break in 20
```

Opmerking: Druk op een toets om te stoppen.

PUNTEN OM TE ONTHOUDEN
Er zijn enkele verschillen tussen INKEY$ en INPUT:
1. INKEY$ geeft geen '?' op het scherm.
2. INKEY$ wacht niet op het intoetsen. Of het bijbehorende teken van de toets, óf de nulstring wordt gegeven als u INKEY$ laat uitvoeren.
3. INKEY$, in tegenstelling tot INPUT, toont niet het gegeven teken.
4. INKEY$ herkent TAB- en DEL-toetsen.
5. In tegenstelling tot INPUT, plaatst INKEY$ u niet in de tekstmodus wanneer deze opdracht wordt uitgevoerd in de grafische modus.
6. INKEY$ geeft maar één teken, terwijl INPUT een hele regel met tekens kan geven.

FOUTEN
Er treedt een 'Type Mismatch'-foutmelding op indien INKEY$ wordt gelijkgesteld aan een numerieke variabele.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
INPUT$
Sectie 1, Hoofdstuk 10: Interactief programmeren

INP Invoer vanuit een poort

BESCHRIJVING

Deze functie geeft de byte die werd gelezen vanaf een bepaalde poort. Het poortnummer moet liggen tussen 0 en 255. Alles boven 255 wordt niet door MSX gebruikt.
Gebruik deze opdracht niet voor commerciële software. Gebruik BIOS - dat is eenvoudiger.

SYNTAX

<num var>=INP(<poortnummer>)

VOORBEELDEN

```
PRINT INP(1)
```

PUNTEN OM TE ONTHOUDEN

INP is het tegenovergestelde van OUT.

FOUTEN

Tenzij u exact weet wat u doet, moet u deze opdracht trachten te vermijden.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

OUT
WAIT

INPUT

BESCHRIJVING
INPUT biedt de mogelijkheid om getallen en strings in te voeren en toe te kennen aan variabelen terwijl het programma loopt, zodat u het programma interactief met de gebruiker kunt maken.
De INPUT-opdracht stopt het programma en wacht op de benodigde invoer vanaf het toetsenbord.
Het is mogelijk om een tekstregel mee te geven aan INPUT; deze wordt geplaatst voor het '?'-teken op het scherm.
Ieder aantal numerieke variabelen of stringvariabelen kan worden ingevoerd in dezelfde opdracht, de een na de ander, zolang ze maar zijn gescheiden door komma's. Er dient echter wel voor te worden gezorgd dat de soort invoer correspondeert met de soort variabele in de INPUT. Gebeurt dit niet dan resulteert dit in een '?Redo from start'-boodschap op het scherm. U moet dan vanaf de allereerste variabele opnieuw invoeren.

SYNTAX
INPUT <lijst van numerieke variabelen of string-variabelen, gescheiden door komma's>
INPUT "boodschap";<lijst met numerieke variabelen of stringvariabelen, gescheiden door komma's>

VOORBEELDEN

```
10 INPUT "GEEF UW NAAM";N$
20 INPUT "UW LEEFTIJD EN GELOOF;A%,R$
30 PRINT N$
40 PRINT A%
50 PRINT R$
```

```
RUN
GEEF UW NAAM? GODZILLA
UW LEEFTIJD EN GELOOF? 12,BOEDDHIST
GODZILLA
12
BOEDDHIST
```

PUNTEN OM TE ONTHOUDEN
In de grafische modus brengt INPUT u terug in de tekstmodus. De INPUT-opdracht antwoordt op drie manieren, als de invoer niet is zoals die zou moeten zijn:

1. 'Type Mismatch'-fout geeft '?Redo from start'. Alles vanaf het begin dient opnieuw te worden ingevoerd.

2. Teveel invoer levert op '?Extra ignored' en de uitvoering van het programma gaat door.
3. Te weinig invoer geeft '??' en de computer blijft wachten op aanvullende invoer.

De enige manier om INPUT te verlaten is door <CTRL> <C> of <CTRL> <STOP> in te drukken. Het is mogelijk om terug te keren in het programma na een dergelijke onderbreking met behulp van CONT, maar de invoer moet dan wel overnieuw gebeuren.
Wordt alleen de <RETURN>-toets ingedrukt dan blijft de waarde in de variabele ongewijzigd.
Iedere variabele dient te zijn gescheiden van de volgende met een komma. De invoerelementen dienen ook te worden gescheiden door komma's.

FOUTEN
Het intikken bij INPUT wordt door de computer behandeld zoals hierboven is uitgelegd.
Is de lengte van een invoerstring meer dan 200 tekens, dan volgt een 'Out of string space'-foutmelding, tenzij u de string-werkruimte vergroot met CLEAR.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
INKEY$
INPUT$
INPUT#
LINE INPUT
Sectie 1, Hoofdstuk 3: Commandomode
Sectie 2, Hoofdstuk 10: Interactief programmeren

INPUT#   Invoer van bestanden

BESCHRIJVING
Deze opdracht stelt u in staat om gegevens van randapparaten in te lezen, in plaats van het toetsenbord, zoals bijvoorbeeld van een cassette.
Voordat deze opdracht kan worden uitgevoerd dient er een speciaal kanaal (channel) te worden geopend met de OPEN-opdracht en een toegekend bestandsnummer.
Het invoerformaat moet exact gelijk zijn aan dat bij gebruik van het toetsenbord.
Gedurende de invoer van numerieke variabelen worden voorafgaande spaties, 'wagen terug'-tekens en nieuwe regel-tekens genegeerd. Hetzelfde geldt voor stringvariabelen. Vraagtekens worden ook genegeerd.

SYNTAX
INPUT#<bestandsnummer>,<lijst met stringvariabelen of numerieke variabelen>

VOORBEELDEN
```
INPUT#1,A$
```

PUNTEN OM TE ONTHOUDEN
Om een bestand te openen voor een cassetterecorder kunt u het volgende uitvoeren:

```
OPEN "CAS:" FOR INPUT AS#1
```

FOUTEN
Wordt een bestandsnummer meegegeven van een nog niet geopend bestand dan volgt de 'Bad File Number'-foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
OPEN
CLOSE
INPUT
INPUT$
Sectie 2, Hoofdstuk 19: Hoe bestanden worden gebruikt

INPUT$

BESCHRIJVING
Deze functie geeft een string met een gegeven aantal tekens, gelezen vanaf het toetsenbord. Geen van deze tekens zal op het beeldscherm worden getoond. <CTRL> <C> beëindigt de uitvoering van de INPUT$-functie.

SYNTAX
<string var>=INPUT$(<num const>)

VOORBEELDEN
```
A$=INPUT$(5)
```

PUNTEN OM TE ONTHOUDEN
In tegenstelling tot de INKEY$-functie wacht INPUT$ totdat een teken wordt ingetoetst. Ook kunnen er meer tekens per keer worden ingevoerd.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
INKEY$
INPUT$(#)
INPUT
Sectie 1, Hoofdstuk 10: Interactief programmeren

INPUT$(#)

BESCHRIJVING
Deze functie geeft een string met een gegeven aantal tekens, gelezen van een ander medium dan het toetsenbord. Geen van de tekens zal op het beeldscherm worden getoond. Het bestandsnummer moet zijn opgegeven en een bestand kan worden geopend met OPEN.

SYNTAX
<string var>=INPUT$(<num const>,<bestandsnummer>)
<string var>=INPUT$(<num const>,#<bestandsnummer>)

VOORBEELDEN
```
W$=INPUT$(5,#1)
```

PUNTEN OM TE ONTHOUDEN
Om een bestand van de cassetterecorder te openen kunt u het volgende uitvoeren:

OPEN "CAS:" FOR INPUT AS#1

FOUTEN
Is het aangeroepen bestand niet geopend dan volgt een 'Bad File Number'-foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
OPEN
CLOSE
INPUT
INPUT#
INPUT$
Sectie 2, Hoofdstuk 19: Hoe bestanden worden gebruikt

INSTR In een string

BESCHRIJVING

INSTR zoekt naar een gegeven string binnen een string en geeft de positie binnen deze string aan indien een gelijkheid wordt gevonden.
INSTR(A$,B$) ... A$ is de hoofdstring en B$ is de te localiseren string.
Wordt geen gelijkheid gevonden dan geeft INSTR 0 terug. Het zoeken begint vanaf de eerste positie van de string, maar u kunt ook zelf de beginpositie aangeven. Deze positiewaarde moet liggen tussen 1 en 255.

SYNTAX

<numeriek>=INSTR(<string var>,"<string>")
<numeriek>=INSTR(<string var>,<string var>)
<numeriek>=INSTR(<numeriek>,<string var>,<string var>)

VOORBEELDEN

```
10 PRINT "HOOFDSTRING IS ALS VOLGT:"
20 PRINT "LIESJE LEERDE LOTJE LOPEN LANGS DE
LANGE LINDELAAN"
30 A$="LIESJE LEERDE LOTJE LOPEN LANGS DE
LANGE LINDELAAN"
40 INPUT "VOER IN WELK TEKEN U WILT LOKALISEREN";B$
50 C%=0
60 C%=INSTR(C%+1,A$,B$)
70 IF C%=0 THEN END
80 PRINT "POSITIE ";C%;" IS ";B$
90 GOTO 60
```

```
RUN
HOOFDSTRING IS ALS VOLGT:
LIESJE LEERDE LOTJE LOPEN LANGS DE LANGE
LINDELAAN
VOER IN WELK TEKEN U WILT LOCALISEREN? L
POSITIE 1 IS L
POSITIE 8 IS L
POSITIE 15 IS L
POSITIE 21 IS L
POSITIE 27 IS L
POSITIE 36 IS L
POSITIE 42 IS L
POSITIE 47 IS L
```

PUNTEN OM TE ONTHOUDEN

U krijgt 0 op INSTR(Z,X$,Y$) in de volgende gevallen:

1. Als Y$ niet wordt gevonden binnen X$.
2. Als Z groter is dan de lengte van X$.
3. Als X$ een nulstring is, dus X$="".
4. Als X$ en Y$ beide nulstrings zijn.

Probeert u de nulstring te lokaliseren dan krijgt u 1 op INSTR.

FOUTEN
Is de zoekpositie 0 of meer dan 255 dan volgt een 'Illegal Function Call'-foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
LEFT$
RIGHT$
MID$
LEN
Sectie 1, Hoofdstuk 14: Het manipuleren van strings

INT Integer

BESCHRIJVING
INT geeft het integere deel van een reëel getal. Het reële getal wordt afgerond naar beneden, tot de dichtstbij gelegen gehele waarde, zowel voor een positief als een negatief getal.

SYNTAX
INT(<num const>)
INT(<num var>)
INT(<num uitdrukking>)

VOORBEELDEN

```
PRINT INT(2.76)
 2
PRINT INT(10.1111)
 10
PRINT INT(-1.234)
-2
```

PUNTEN OM TE ONTHOUDEN
Let er op dat de INT van een negatief getal de dichtstbij gelegen lagere integer (dus kleinere) waarde geeft, bijvoorbeeld INT(-0.2) geeft -1.
INT verschilt van FIX. FIX geeft de dichtstbij gelegen lagere integer voor een positief argument en de dichtstbij gelegen hogere integer voor een negatief argument.

FOUTEN
Wordt INT toegepast op strings dan treedt een 'Type Mismatch'-foutmelding op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
FIX
Sectie 1, Hoofdstuk 4: Meer over rekenen
Sectie 1, Hoofdstuk 5: Functies

INTERVAL ON/OFF/STOP

BESCHRIJVING

MSX-BASIC 'timet' zichzelf en bij specifieke tijdsintervallen kunt u een bepaalde routine laten uitvoeren met behulp van de ON INTERVAL-opdracht (zie ON INTERVAL).

INTERVAL ON/OFF/STOP activeert of de-activeert de ON INTERVAL-evenementdetectie.

Om ON INTERVAL GOSUB te gebruiken dient u deze te activeren met INTERVAL ON. Na INTERVAL ON controleert de computer na het uitvoeren van elke regel of het tijd is geworden om de ON INTERVAL GOSUB-routine aan te roepen.

INTERVAL OFF laat de computer weten te stoppen met de controle op verstreken tijd en de-activeert de ON INTERVAL GOSUB.

INTERVAL STOP geeft de computer aan om geen ON INTERVAL GOSUB uit te voeren, maar om te wachten met de uitvoering als de opgegeven tijd is verstreken, totdat een INTERVAL ON wordt uitgevoerd.

SYNTAX

INTERVAL ON
INTERVAL OFF
INTERVAL STOP

VOORBEELDEN

Druk de <J>-toets in om de interval-detectie te activeren. Druk op de <N>-toets om de interval-detectie te de-activeren. U krijgt biep te horen als de interval-interruptie wordt geactiveerd.

```
10 ON INTERVAL=60 GOSUB 50
20 IF INKEY$="J" THEN INTERVAL ON
30 IF INKEY$="N" THEN INTERVAL OFF
40 GOTO 20
45 REM INTERVAL SUBROUTINE
50 BEEP
60 RETURN
```

REGEL 10 zet de intervaltijd op 1 seconde en de subroutine vanaf regel 50.
REGEL 20 activeert de interruptie.
REGEL 30 de-activeert de interruptie.
REGEL 40 oneindige lus met regel 20.
REGEL 60 keert terug, naar de plaats van de aanroep in het hoofdprogramma.

PUNTEN OM TE ONTHOUDEN

De tijdsinterval wordt gezet in eenheden van 1/60 seonde.

De interrupties worden gede-activeerd in de commandomodus, dit is als het

programma niet actief is en gedurende foutafhandelingsroutines.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ON INTERVAL GOSUB
Sectie 2, Hoofdstuk 9: Evenementbehandeling en interruptie door BASIC

KEY

BESCHRIJVING
Dit commando wordt gebruikt om een functie toe te kennen aan een functietoets. Iedere MSX-computer beschikt over vijf functietoetsen, waarbij iedere toets twee functies voorstelt. De eerste functie wordt verkregen door het indrukken van de betreffende toets en de tweede functie door zowel de betreffende toets als <SHIFT> in te drukken.
Als de computer wordt aangeschakeld zijn de toetsen reeds als volgt geïnitialiseerd:

F1	= COLOR (s)	kleur
F2	= AUTO (s)	AUTO
F3	= GOTO (s)	GOTO
F4	= LIST (s)	LIST
F5	= RUN (r)	start programma
F6	= COLOR 15,4,7 (r)	vervangingskleuren
F7	= CLOAD"	laden van cassette
F8	= CONT (r)	zet het programma voort
F9	= LIST. (r)(u)(u)	toont laatst gerefereerde regel en verplaatst de cursor naar het begin van die regel
F10	= (CLS) RUN (r)	wist het scherm en start het programma

(s)	=	spatie
(r)	=	RETURN
(CLS)	=	maak scherm schoon
(u)	=	cursor één regel omhoog

U kunt elk van de functietoetsen een nieuwe functie geven met de KEY-opdracht. De string dient echter hooguit 15 tekens lang te zijn.
Wilt u een functie definiëren die direct wordt uitgevoerd na het indrukken van de functietoets, voeg dan een 'wagen terug'-teken toe aan het einde van de string (dus een CHR$(13)). U kunt ook andere besturingstekens toevoegen, zoals het CLS-commando.

SYNTAX
KEY<num const>,<string>
KEY<num const>,<string-uitdrukking>
waarin <num const> moet liggen tussen 1 en 10.

VOORBEELDEN

`KEY2,"PRINT FRE(0)"+CHR$(13) ...`	drukt aantal vrije bytes in BASIC af
`KEY4,"RENUM"+CHR$(13)`	opnieuw nummeren
A$="KEY LIST"	
KEY5,A$+CHR$(13)	geeft 'KEY LIST' (r)

Opmerking: CHR$(13) is het besturingsteken voor de <RETURN>-toets.

PUNTEN OM TE ONTHOUDEN
Als de machine wordt aangeschakeld, wordt de inhoud van de functietoetsen onderaan het scherm getoond. Wilt u hier van af, gebruik dan de KEY OFF-opdracht.
Onthoud goed dat besturingstekens kunnen worden opgenomen in de functietoetsdefinitie, zodat u het scherm kunt wissen of een BEEP kunt genereren enz.

FOUTEN
Vergeet de aanhalingstekens niet rondom de string.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
KEY ON/OFF
KEY LIST
Sectie 1, Hoofdstuk 8: De functietoetsen

KEY LIST

BESCHRIJVING
Deze opdracht laat een overzicht zien van de inhoud per functietoets, zonder verder iets uit te voeren.
Alle besturingstekens worden veranderd in spaties als ze worden getoond.

SYNTAX
KEY LIST

VOORBEELDEN

```
KEY LIST
color
auto
goto
list
run
color 15,7,7
cload"
cont
list.
run
```

PUNTEN OM TE ONTHOUDEN
Zie ook de KEY-opdracht, waarin wordt verteld hoe de functies per functietoets kunnen worden gewijzigd.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
KEY
KEY ON/OFF
Sectie 1, Hoofdstuk 8: De functietoetsen

KEY ON/OFF

BESCHRIJVING
Als de computer wordt aangeschakeld laat de 24e regel op het scherm zien wat de inhoud van elk der functietoetsen is. Dit kan nuttig zijn bij het programmeren, maar het kan ook in de weg staan wanneer u een programma uitvoert. De KEY ON/OFF-functie stelt u staat om het overzicht van de 24e regel wel of niet te tonen.

SYNTAX
KEY ON
KEY OFF

VOORBEELDEN
```
KEY ON
KEY OFF
```

PUNTEN OM TE ONTHOUDEN
KEY LIST geeft een overzicht van de inhoud per functietoets.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
KEY
KEY LIST
Sectie 1, Hoofdstuk 8: De functietoetsen

KEY () ON/OFF/STOP

BESCHRIJVING

De KEY () ON/OFF/STOP-opdracht activeert/de-activeert de detectie van een opgegeven functietoets. Zodra de detectie van een functietoets is geactiveerd, zal na het intoetsen van de betreffende functietoets door de computer naar een subroutine worden gesprongen, zoals opgegeven in de ON KEY GOSUB-opdracht.
De individuele functietoetsen kunnen apart worden geactiveerd of gedeactiveerd. Functietoetsen die niet staan gespecificeerd in KEY () ON worden verondersteld de-actief te zijn.
Wordt KEY () STOP uitgevoerd dan zal de computer niet naar de in ON KEY GOSUB opgegeven subroutine springen bij het intoetsen van de betreffende functietoets. Wordt na het intoetsen van de functietoets een KEY () ON uitgevoerd dan springt de computer direct naar de subroutine.
KEY () OFF zal de interruptie definitief afbreken voor de betreffende toets.

SYNTAX

KEY (<num const>) ON
KEY (<num const>) OFF
KEY (<num const>) STOP

VOORBEELDEN

Drukt u de functietoets F0 in, dan hoort u de computer tweemaal een bieptoon geven. Drukt u op F1 dan volgt slechts één bieptoon.

```
10 ON KEY GOSUB 50,60
20 KEY(0) ON
30 KEY(1) ON
40 GOTO 40
50 BEEP
60 BEEP
70 RETURN
```

REGEL 10	zet de subroutine voor F0 op regel 50 en voor F1 op regel 60.
REGEL 20	activeert F0.
REGEL 30	activeert F1.
REGEL 40	oneindige lus die wacht op een functietoets.
REGEL 50	BEEP (F0).
REGEL 60	BEEP (voor beide functietoetsen).
REGEL 70	keert terug naar de plaats van aanroep, hier is dat regel 40.

Let er op dat bovenstaand programma niets uitvoert indien u een andere functietoets indrukt.

PUNTEN OM TE ONTHOUDEN
De functietoets-interruptie wordt gede-activeerd gedurende de foutafhandeling.

FOUTEN
Vergeet niet eerst een ON KEY GOSUB-opdracht uit te voeren. Deze opdracht is volledig verschillend van KEY ON/OFF.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ON KEY GOSUB
Sectie 2, Hoofdstuk 9: Evenementbehandeling en interruptie door BASIC

LEFT$

BESCHRIJVING

LEFT$ is een stringmanipulatie-functie, die de linker tekens van een gegeven string geeft. Is de bronstring korter dan de opgegeven waarde dan wordt de volledige string gegeven.

SYNTAX

LEFT$(<string-var>,<num const>)
LEFT$(<string-var>,<num var>)
LEFT$(<string-var>),<num exp>)
Het aantal tekens moet liggen tussen 0 en 255.

VOORBEELDEN

```
10 A$="JAN JANSEN"
20 P=INSTR(A$," ")
30 B$=LEFT$(A$,P-1)
40 PRINT B$
```

REGEL 20 localiseert de positie van de spatie in A$.
REGEL 30 geeft de linker P-1 tekens van A$ van de voornaam.

```
RUN
JAN
Ok
```

PUNTEN OM TE ONTHOUDEN

Als het aantal opgegeven tekens in het argument nul is dan wordt de nulstring teruggegeven.
Om de achternaam, 'JANSEN', in het voorbeeld te verkrijgen, dient u MID$ te gebruiken en wel als volgt:

```
PRINT MID$(A$,P+1)
```

FOUTEN

Het door het argument aangegeven aantal dient een integer te zijn die ligt tussen 0 en 255; anders volgt een 'Type Mismatch'-foutmelding.
Wordt naar een niet-bestaande variabele verwezen dan treedt een 'Illegal Function Call' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

INSTR, RIGHT$, MID$, LEN
Sectie 1, Hoofdstuk 14: Manipuleren van strings

## LEN		Lengte van een string

BESCHRIJVING
LEN geeft de lengte van een gegeven string.

SYNTAX
LEN(<string>)

VOORBEELDEN

```
10 A$="HALLO"+CHR$(7)
20 B=LEN(A$)
30 PRINT A$
40 PRINT "DE LENGTE VAN A$ IS ";B
```

REGEL 10	CHR$(7) is een BEEP.
REGEL 30	drukt 'HALLO' af en geeft een BEEP.

```
RUN
HALLO                (er volgt een bieptoon)
DE LENGTE VAN A$ IS 6
```

PUNTEN OM TE ONTHOUDEN
Deze functie geeft altijd een integere waarde terug.

FOUTEN
De functie geeft 0, indien verwezen wordt naar een niet-bestaande string-variabele.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
RIGHT$
LEFT$
MID$
INSTR
Sectie 1, Hoofdstuk 15: Functies

LET

BESCHRIJVING
Als u een waarde toekent aan een stringvariabele of een numerieke variabele, dan gebruikt u LET, zoals:

LET A=100

LET is niet verplicht en dient eigenlijk te worden vermeden omdat deze instructie alleen maar geheugenruimte gebruikt. Dus A=100 is identiek aan bovenstaand voorbeeld.

SYNTAX
LET <variabele>=<uitdrukking>

VOORBEELDEN
```
LET A=100
LET W$="WEEK"+"END"
```

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 2: Commandomode

LINE Tekent een lijn

BESCHRIJVING

Deze grafische opdracht tekent lijnen en rechthoeken. De rechthoeken kunnen worden ingekleurd met een willekeurige kleur.

SYNTAX

LINE <coördinaten-specificatie>

Tekent een lijn in de huidige voorgrondkleur, vanaf het laatst verwezen punt tot het punt dat wordt aangegeven door de <coördinaten-specificatie>.

LINE <coördinaten-specificatie>,<kleur>

Tekent een lijn in de aangegeven kleur, vanaf het laatst verwezen punt tot het punt dat wordt aangegeven door de <coördinaten-specificatie>.

LINE <coördinaten-specificatie>-<coördinaten-specificatie>

Tekent een lijn van het punt dat wordt aangegeven door de eerste <coördinaten-specificatie> tot het punt dat wordt aangegeven door de tweede <coördinaten-specificatie>, in de huidige voorgrondkleur.

LINE <coördinaten-specificatie>-<coördinaten-specificatie>,<kleur>

Tekent een lijn van het punt dat wordt aangegeven door de eerste <coördinaten-specificatie> tot het punt dat wordt aangegeven door de tweede <coördinaten-specificatie> in de opgegeven kleur.

LINE <coördinaten-specificatie>-<coördinaten-specificatie>,<kleur>,B

Tekent een rechthoek in de aangegeven kleur, waarbij de linkerbovenhoek wordt aangegeven door de eerste <coördinaten-specificatie>, de rechteronderhoek door de tweede <coördinaten-specificatie>.

LINE <coördinaten-specificatie>-<coördinaten-specificatie>,<kleur>,BF

Identiek aan het vorige voorbeeld, maar nu wordt de rechthoek ingekleurd met de aangegeven kleur.

Verklaring van <coördinaten-specificatie>

Er bestaan twee formaten, de één een relatief en de ander een absoluut coördinatenformaat:

1. (<x-coördinaat>,<y-coördinaat>)
   Deze coördinaten specificeren absolute posities op het scherm. x kan liggen tussen 0 en 255 en y tussen 0 en 191.

2. STEP (<x-verplaatsing>,<y-verplaatsing>)
   Deze coórdinaten geven een plaats aan die relatief is ten opzichte van het laatste, door de computer verwezen punt. Valt dit punt buiten het scherm dan tekent de computer tot aan de rand van het scherm.
   <x-verplaatsing> en <y-verplaatsing> kunnen negatief zijn, afhankelijk van de door u gekozen richting.

Opmerking: <x-verplaatsing>, <y-verplaatsing>, <x-coördinaat> en <y-coördinaat> kunnen elk ook variabelen, uitdrukkingen of gewoon constanten zijn.

VOORBEELDEN

```
10 SCREEN 2
20 COLOR 4,1,1
30 CLS
40 LINE (50,50)-(100,100)
50 LINE (50,50)-(100,100),4,B
60 LINE STEP (-50,20)-STEP(50,50),9,BF
70 X=10:Y=10
80 LINE (15*X,10*Y)-STEP(50,20),6,B
90 GOTO 90
```

```
U ZIET EEN VIERKANT MET DIAGONAAL, EEN PAARS
VIERKANT EN EEN RECHTHOEK
```

PUNTEN OM TE ONTHOUDEN

Coördinaten met reële getallen worden afgekort tot integere waarden. De kleurcode moet een geldige waarde zijn die ligt tussen 0 en 15 (zie COLOR).

FOUTEN

De coördinaten mogen groter of kleiner zijn dan de grootte van het beeldscherm. Valt een waarde buiten de toegestane marge dan volgt een 'Overflow error'-foutboodschap (bijv. -100000,100000 levert 'Overflow error' op).

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

PAINT
COLOR
DRAW
Sectie 1, Hoofdstuk 24: Het tekenen van lijnen en vierkanten
Sectie 2, Hoofdstuk 13: Geavanceerde grafische toepassingen II: kleuren in grafische modus 2 met hoog oplossend vermogen

LINE INPUT

BESCHRIJVING
Deze opdracht is identiek aan INPUT, maar stelt u in staat een hele regel tot een lengte van 200 tekens in te voeren, inclusief komma's.
Ook kan tekst worden meegegeven, zoals bij INPUT, maar een vraagteken zal niet automatisch worden afgedrukt, in tegenstelling tot INPUT.

SYNTAX
LINE INPUT <string-var>
LINE INPUT "<tekst>",<string-var>

VOORBEELDEN

```
10 LINE INPUT "WAT ZAL IK NU DOEN?";S$
20 PRINT "OK, ";S$
```

```
RUN
WAT ZAL IK NU DOEN? PAK DE SLEUTEL EN HET ZWAARD
OK, PAK DE SLEUTEL EN HET ZWAARD
```

PUNTEN OM TE ONTHOUDEN
U kunt LINE INPUT onderbreken met <CTRL> <C> of <CTRL> <STOP>. CONT brengt u terug in LINE INPUT.
Deze opdracht kan niet worden uitgevoerd in grafische modes. U dient eerst naar de tekstmodus terug te keren.
Het maximale aantal tekens in een regel is 200. Deze limiet wordt bepaald door de grootte van het string-werkgebied in de systeem-RAM. Wilt u deze ruimte vergroten, gebruik dan CLEAR.

FOUTEN
Deze opdracht kan alleen met een stringvariabele worden uitgevoerd. Probeert u een numerieke variabele dan volgt een 'Type Mismatch'-foutboodschap.
Als er een te grote string wordt ingevoerd in het string-werkgebied volgt een 'Out of String Space'-foutmelding. (De vervangingswaarde biedt ruimte aan maximaal 200 tekens.)

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
LINE INPUT #
INPUT
Sectie 1, Hoofdstuk 10: Interactief programmeren

LINE INPUT#

BESCHRIJVING
Deze opdracht is gelijk aan INPUT#, maar stelt u daarbij in staat een hele regel te lezen van 200 tekens, vanuit een sequentieel bestand op een extern medium, zoals een cassette.
In principe komt het er op neer dat u spaties en komma's, zowel als tekens vanaf een cassette in een string kunt lezen.

SYNTAX
LINE INPUT #<bestandsnummer>,<string-var>
<bestandsnummer> is het nummer van het geopende bestand.

VOORBEELDEN
```
LINE INPUT#1,A$
```

PUNTEN OM TE ONTHOUDEN
LINE INPUT# leest alles tot aan het 'wagen terug'-symbool. Het is nuttig wanneer iedere regel in het bestand in velden is afgebroken.
Ook kan een BASIC-programma, dat stond opgeslagen in ASCII-formaat, in een programma worden ingelezen met behulp van LINE INPUT#.

FOUTEN
U kunt voor dit commando alleen stringvariabelen gebruiken. Als u een numerieke variabele probeert dan volgt een 'Type Mismatch'-foutboodschap.
Als de ingevoerde string te groot wordt om in het string-werkgebied te worden opgeslagen dan volgt een 'Out of String Space'-foutboodschap. (De vervangingswaarde biedt ruimte tot maximaal 200 tekens.)

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
INPUT
INPUT#
INPUT$(#)
OPEN
LINE INPUT
Sectie 2, Hoofdstuk 19: Hoe bestanden worden gebruikt

LIST

BESCHRIJVING

Deze opdracht toont de BASIC-programmalijst, zoals die is opgeslagen in het geheugen van de computer. U kunt het gehele programma, maar ook gedeelten tonen.

SYNTAX

LIST	Toont het gehele programma.
LIST <regel>	Toont de aangegeven regel.
LIST <regel1>-<regel2>	Toont het programma vanaf <regel1> tot <regel2>.
LIST <regel>-	Toont het programma vanaf <regel>.
LIST -<regel>	Toont het programma vanaf het begin tot aan <regel>.
LIST.	Toont de door de computer als laatst verwezen regel.

VOORBEELDEN

`LIST 100-200`	Lijst van regel 100 tot en met regel 200.
`LIST -100`	Lijst vanaf het begin tot en met regel 100.

PUNTEN OM TE ONTHOUDEN

Functietoets F4 is gedefinieerd als 'LIST', zodra de computer wordt aangeschakeld.

Alle kleine letters in de lijst, behalve die tussen aanhalingstekens, worden hoofdletters.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

LLIST

Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's

Sectie 2, Hoofdstuk 1: Geavanceerde programmabewerking

LLIST

BESCHRIJVING
Dit commando drukt de BASIC-programmalijst af op de printer. U kunt het gehele programma, maar ook een gedeelte ervan afdrukken.

SYNTAX

LLIST	Drukt het gehele programma af.
LLIST <regel>	Drukt de aangegeven regel af.
LLIST <regel1>-<regel2>	Drukt het programma af vanaf <regel1> tot en met <regel2>.
LLIST <regel>-	Drukt het programma af vanaf <regel>.
LLIST -<regel>	Drukt het programma af vanaf het begin tot aan <regel>.
LLIST.	Drukt de door de computer als laatst verwezen regel af.

VOORBEELDEN

`LLIST 100-200`	Drukt de regels 100 tot en met 200 af.
`LLIST -100`	Drukt de regels vanaf het begin tot en met 100 af.

PUNTEN OM TE ONTHOUDEN
Alle kleine letters, behalve die tussen aanhalingstekens, worden als hoofdletters afgedrukt in de programmalijst.

FOUTEN
Als er geen printer is aangesloten, gebeurt er niets.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
LIST
Sectie 2, Hoofdstuk 23: Randapparatuur

LOAD

BESCHRIJVING

Dit commando laadt een BASIC-bestand dat staat opgeslagen in ASCII-formaat vanaf een gespecificeerd medium.
Op dit moment is alleen de cassette-mogelijkheid geïmplementeerd.
LOAD beschikt over de mogelijkheid het programma automatisch uit te voeren nadat het is geladen.

SYNTAX

LOAD"<naam van medium>"	Laadt het eerste BASIC-bestand in ASCII-formaat dat wordt gevonden.
LOAD"<naam van medium><bestandsnaam>"	Laadt het BASIC-programma met de opgegeven naam.
LOAD"<naam van medium><bestandsnaam>",R	Laadt het BASIC-bestand met de aangegeven naam en voert dit daarna automatisch uit.

<naam van medium> = CAS: (momenteel).

VOORBEELDEN

```
LOAD "CAS:SPEL",R
```

Dit commando laadt het bestand met de naam 'SPEL', dat staat opgeslagen in ASCII-formaat en voert dit direct daarna automatisch uit.

PUNTEN OM TE ONTHOUDEN

Als LOAD wordt uitgevoerd sluit de computer alle nog openstaande bestanden af en start met het laden van het nieuwe BASIC-bestand, waarbij de huidige geheugeninhoud wordt overschreven.
<CTRL> <Z> wordt behandeld als EOF, dit is het einde van een bestand.

FOUTEN

Machinecode kan niet worden geladen met LOAD; hiervoor dient u BLOAD te gebruiken.

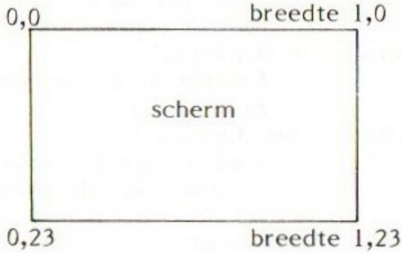
BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

SAVE
CLOAD
CSAVE
BLOAD
BSAVE
Sectie 2, Hoofdstuk 2: Het laden en bewaren van bestanden op cassette.

LOCATE

BESCHRIJVING

LOCATE beweegt de cursor naar een opgegeven positie. Ook kan hiermee het al dan niet verschijnen van de cursor worden geregeld. Deze opdracht wordt gewoonlijk samen met PRINT-opdrachten gebruikt om op een specifieke plaats op het tekstscherm iets af te drukken.
U kunt de cursor verplaatsen met LOCATE naar iedere plaats op het scherm. De enige beperking is dat binnen de grenzen van de tekstmodus of WIDTH moet worden gewerkt.



<- breedte vastgesteld ->
door WIDTH-instructie,
of door vervangingswaarde
(hetgeen afhangt van de tekstmodus)

De vervangingswaarde van beide tekstmodi is als volgt:
MODE 0: WIDTH = 37 (maximum breedte = 40)
MODE 1: WIDTH = 29 (maximum breedte = 32)

SYNTAX

LOCATE <x-coördinaat>,<y-coördinaat>	Beweegt de cursor naar de opgegeven coördinaten.
LOCATE <x-coördinaat>,<y-coördinaat>,0	Beweegt de cursor naar de opgegeven coördinaten, maar de-activeert de cursor.
LOCATE <x-coördinaat>,<y-coördinaat>,1	Beweegt de cursor naar de opgegeven coördinaten en activeert de cursor.

<x-coördinaat> en <y-coördinaat> kunnen beide variabelen, constanten of uitdrukkingen zijn, maar moeten liggen binnen de grenzen van het scherm. Alle reële getallen worden omgezet in integere waarden.

VOORBEELDEN

```
10 FOR Y=0 TO 6
20 FOR X=0 TO 20
30 LOCATE X,Y
40 PRINT "E"
50 NEXT:NEXT
60 PRINT "HEEL VEEL KEREN E"
```

```
EEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEE
HEEL VEEL KEREN E
```

PUNTEN OM TE ONTHOUDEN

LOCATE werkt niet in de grafische modus.

LOCATE kan worden gebruikt om de positie van een INPUT-tekst te lokaliseren.

Er bestaat een soortgelijke functie, TAB, die wordt gebruikt in de vorm van PRINT TAB, doch deze is veel meer beperkt en biedt alleen mogelijkheid tot verplaatsing in de x-richting, vandaar dat LOCATE vaker wordt toegepast.

FOUTEN

Overtuig u ervan dat de coördinaten niet buiten de schermgrenzen vallen; er volgt anders een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

PRINT
INPUT
TAB
Sectie 1, Hoofdstuk 9: Meer over PRINT en het beeldscherm

LOG

BESCHRIJVING
LOG geeft de natuurlijke logaritme van een gegeven getal in dubbele nauwkeurigheid.

SYNTAX
LOG(<num const>)
LOG(<num var>)
LOG(<num uitdrukking>)

VOORBEELDEN
```
PRINT LOG(10)
 2.302585092994
```

PUNTEN OM TE ONTHOUDEN
Met LOG bepaalt men de natuurlijke logaritme met basis e. Dit is een andere logaritme dan die met basis 10.
Het argument moet groter zijn dan nul.

FOUTEN
Als een negatief argument wordt meegegeven dan volgt een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
EXP
Sectie 1, Hoofdstuk 19: Wiskundige functies

LPOS Positie van een regel

BESCHRIJVING
LPOS geeft de positie van de printerkop, zoals staat aangegeven in de printerbuffer.
LPOS heeft een dummy-variabele X nodig.
LPOS hoeft niet de fysieke positie voor te stellen van de printerkop.

SYNTAX
LPOS(X)

VOORBEELDEN
```
PRINT LPOS(X)
```
of
```
A=LPOS(X)
```

PUNTEN OM TE ONTHOUDEN
U kunt geen waarde toekennen aan deze systeemvariabele.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 2, Hoofdstuk 23: Randapparatuur

LPRINT

BESCHRIJVING
LPRINT is het afdrukcommando voor de printer en is gelijk aan PRINT.

SYNTAX
LPRINT

VOORBEELDEN
`LPRINT "Dit is een test";123`
zal op de printer '`Dit is een tekst 123`' afdrukken

PUNTEN OM TE ONTHOUDEN
Denk eraan dat er een printer moet zijn aangesloten!

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
PRINT
LPRINT USING
Sectie 2, Hoofdstuk 23: Randapparatuur

LPRINT USING

BESCHRIJVING
LPRINT USING is identiek aan PRINT USING, behalve dat deze op een printer afdrukt. Zie PRINT USING voor meer details.

SYNTAX
Zie PRINT USING

VOORBEELDEN
```
LPRINT "IK BEN GEK OP MSX"
```
zal op de printer afdrukken:
```
'IK BEN GEK OP MSX'
```

PUNTEN OM TE ONTHOUDEN
Vergeet niet eerst een printer aan te sluiten, alvorens u dit commando uitvoert.

FOUTEN
Zie PRINT USING.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
LPRINT
PRINT USING
Sectie 2, Hoofdstuk 8: PRINT USING
Sectie 2, Hoofdstuk 23: Randapparatuur

MAXFILES Maximale aantal bestanden

BESCHRIJVING
Deze opdracht definieert het maximale aantal bestanden. De waarde kan liggen tussen 0 en 15. Wordt MAXFILES niet gezet dan veronderstelt de computer MAXFILES=1. Is MAXFILES=0 dan kan alleen SAVE en LOAD worden uitgevoerd.

SYNTAX
MAXFILES=<num const>
MAXFILES=<num var>
MAXFILES=<num uitdrukking>

VOORBEELDEN
```
1000 MAXFILES=5
```

PUNTEN OM TE ONTHOUDEN
MAXFILES leidt tot een aanpassing van het bestandsbesturingsblok in het geheugen. Het besturingsblok wordt gebruikt bij het benaderen van randapparatuur met bestanden.

FOUTEN
Vaak wordt de S vergeten in MAXFILES.
Valt het argument buiten de toegestane waarden dan volgt een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
OPEN
CLOSE
Sectie 2, Hoofdstuk 19: Hoe bestanden worden gebruikt

MERGE Samenvoegen van twee bestanden

BESCHRIJVING

Dit commando biedt de mogelijkheid om twee BASIC-programma's samen te voegen tot één programma.

Gesteld u heeft twee bestanden, PROG1 en PROG2, en u wilt PROG2 toevoegen aan PROG1, zodat PROG2 na PROG1 komt bij de uiteindelijke MERGE. Laten we even aannemen dat beide programma's op cassette staan.

Allereerst laden we PROG2 en hernummeren dit programma, zodat de regelnummers groter zijn dan die in PROG1. Sla dit programma op in een ASCII-bestand (met SAVE).

Laad nu PROG1 van de cassette en tik in: MERGE"PROG2" en start de cassetterecorder. De computer voegt nu PROG2 toe aan het einde van PROG1.

Blijkt een regelnummer in beide programma's voor te komen dan wordt deze vervangen door de regel die wordt toegevoegd.

SYNTAX

MERGE"<randapparaat>"
MERGE"<randapparaat><bestandsnaam>"
<randapparaat>=CAS: voor de cassette.

VOORBEELDEN

Hoe worden twee programma's samengevoegd?

PROG1	PROG2
10 PRINT "HALLO"	10 FOR I=1 TO 10
20 PRINT "WELKOM"	20 PRINT "MSX"
30 PRINT "BIJ UW"	30 NEXT I

CLOAD PROG2 en RENUM 50 zodat na LIST het volgende verschijnt:

```
50 FOR I=1 TO 10
60 PRINT "MSX"
70 NEXT I
```

SAVE PROG2 in ASCII-formaat met SAVE"CAS:PROG2".
Laad vervolgens PROG1 met CLOAD.
MERGE"CAS:PROG2" voegt PROG2 toe aan het einde van PROG1.
LIST laat uiteindelijk het volgende zien:

```
10 PRINT "HALLO"
```

```
20 PRINT "WELKOM"
30 PRINT "BIJ UW"
50 FOR I=1 TO 10
60 PRINT "MSX"
70 NEXT I
```

PUNTEN OM TE ONTHOUDEN

Bestaat een regelnummer in beide programma's dan zal de regel van het programma dat wordt toegevoegd uiteindelijk overblijven.
Wordt de naam van een bestand weggelaten dan zal het volgende ASCII-bestand dat zich op de cassette bevindt worden geladen met MERGE.
<CTRL> <Z> (EOF) vertelt de computer dat het einde van het bestand is bereikt.

FOUTEN

Zorg dat het programma waarmee u MERGE uitvoert met de correcte naam wordt aangeduid, omdat u dit anders misloopt.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

SAVE
Sectie 2, Hoofdstuk 11: Het bewaren en laden via de cassetterecorder

MID$

BESCHRIJVING
Deze functie geeft het midden van een string.

SYNTAX

MID$(<string-var>,x,y)	Geeft het middengedeelte van de string, y tekens lang, vanaf positie x.
MID$(<string-var>,x)	Geeft de rest van de string vanaf positie x.

VOORBEELDEN
MID$ wordt vaak toegepast bij het scheiden van woorden in zinnen. Hier is een goed voorbeeld:

```
10 A$="MARTIN LUTHER KING"
20 B1=INSTR(A$," ")
30 B2=INSTR(B1+1,A$," ")
40 PRINT LEFT$(A$,B1-1)
50 PRINT MID$(A$,B1+1,B2-B1-1)
60 PRINT MID$(A$,B2+1)
```

```
RUN
MARTIN
LUTHER
KING
```

PUNTEN OM TE ONTHOUDEN
x en y moeten liggen tussen 1 en 255.
Als x groter is dan de lengte van de gegeven string dan geeft MID$ de nulstring terug.

FOUTEN
MID$ is een stringfunctie. Vermijd 'Type Mismatch'-fouten.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
RIGHT$
LEFT$
LEN
INSTR
Sectie 1, Hoofdstuk 14: Het manipuleren van strings

MOD MODulus

BESCHRIJVING
Deze functie voert modulus rekenkundige bewerkingen uit. Dit betekent dat de bewerker MOD de rest geeft van een deling met integere waarden.

10 MOD 3=1

omdat 10/3 gelijk is aan 3 en de rest is 1.

SYNTAX
<num var>=<numeriek>MOD<numeriek>

VOORBEELDEN
```
10 PRINT 15 MOD 5
```
en u krijgt 0.

PUNTEN OM TE ONTHOUDEN
De operands worden afgekort tot integere waarden voor de bewerking.

FOUTEN
Als strings worden meegegeven volgt een 'Type Mismatch'-foutboodschap.
Als het argument buiten de toegestane waarden valt (-32768 tot 32767) dan volgt 'Overflow Error'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
(DIV)
Sectie 2, Hoofdstuk 4: Uitdrukkingen en bewerkers.

MOTOR

BESCHRIJVING

Deze opdracht stelt u in staat om de cassetterecorder te gebruiken met 'REMOTE CONTROL'. Hiervoor moet de cassetterecorder uiteraard over een dergelijke aansluiting beschikken. Deze ingang dient te worden verbonden met de 'REMOTE CONTROL'-aansluiting van de computer.
MOTOR verwijst naar de motor van de cassetterecorder.

SYNTAX

MOTOR	Verwisselt de motorschakelaar, dus AAN indien UIT en UIT indien AAN.
MOTOR ON	Schakelt de motor aan.
MOTOR OFF	Schakelt de motor uit.

PUNTEN OM TE ONTHOUDEN

Dit commando is alleen nuttig voor cassetterecorders die over een 'REMOTE CONTROL'-aansluiting beschikken.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Zie het gedeelte over de cassette.

NEW

BESCHRIJVING
NEW verwijdert het gehele programma uit het computergeheugen en zet alle variabelen en systeemvariabelen terug op nul. Het verwijderde programma kan na een NEW niet worden terugverkregen.
De bedoeling van NEW is het geheugen gereed te maken voor een nieuw programma.

SYNTAX
NEW

VOORBEELDEN
NEW

PUNTEN OM TE ONTHOUDEN
NEW maakt het scherm niet schoon.
Toets altijd eerst NEW in, voordat u een nieuw programma gaat laden. Doet u dit niet dan wordt het oude programma gemixed met het te laden programma.

FOUTEN
Fataal, als u per ongeluk een NEW intoetst!

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 3: Het schrijven van een programma

NEXT

BESCHRIJVING
NEXT wordt altijd met FOR gecombineerd in een set van opdrachten en geeft de computer aan, dat dient te worden teruggekeerd naar de corresponderende FOR-opdracht, tenzij de lus aan de laatste iteratie bezig was.
Voor meer informatie over FOR...NEXT-lussen kunt u bij FOR kijken.

SYNTAX
NEXT
NEXT <variabele>
NEXT <variabele>,<variabele>, ... lijst met variabelen

VOORBEELDEN

```
10 FOR I=1 TO 9
20 PRINT I;
30 NEXT I
```

```
RUN
1 2 3 4 5 6 7 8 9
Ok
```

PUNTEN OM TE ONTHOUDEN
<variabele> na NEXT mag worden weggelaten.

FOUTEN
Het nu volgende is een veelgemaakte fout - NEXT I en NEXT J worden verwisseld:

```
10 FOR I=1 TO 5
20 FOR J=1 TO 10
30 PRINT I,J
40 NEXT I
50 NEXT J
```

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
FOR, TO, STEP
Sectie 1, Hoofdstuk 5: Het gebruik van lussen

NOT

BESCHRIJVING
NOT wordt toegepast in FOR...NEXT-conditietests en in BOOLEAN-bewerkingen. Het tegenovergestelde wordt teruggegeven, dus NOT (WAAR) levert (NIET WAAR) op.
De waarheidstabel voor NOT:

NOT	X	NOT X
	0	1
	1	0

SYNTAX
<num var>=NOT(<numeriek>)
IF NOT (<conditie>) THEN

VOORBEELDEN
```
A=0
PRINT NOT(A)
```
geeft -1. Waarom?
Welnu, A=&B0000000000000000 binair. NOT(A) levert op &B1111111111111111, dit is -1 decimaal.
```
IF NOT (A=100 AND B<900) THEN GOTO 1000
```

PUNTEN OM TE ONTHOUDEN
De argumenten moeten liggen tussen -32768 en 32767.
Gedeelten van reële getallen worden afgekapt.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
IF
THEN
ELSE
AND
OR
XOR
IMP
EQV
Sectie 2, Hoofdstuk 6: BOOLEAN algebra
Sectie 2, Hoofdstuk 7: BOOLEAN II: De IF...THEN...ELSE

OCT$ Octale string

BESCHRIJVING

OCT$ levert een string op die de octale weergave is van het decimale argument.
Het octale getallenstelsel is het stelsel dat 8 als grondtal voert. Het octale stelsel gebruikt de cijfers 0, 1, 2, 3, 4, 5, 6, 7 en 8. Iedere positie in het getal stelt een achtste macht voor.
Het argument moet een numerieke uitdrukking zijn die een waarde heeft tussen -32768 en 65535.

SYNTAX

<string-var>=OCT$(<numeriek>)

VOORBEELDEN

```
10 PRINT OCT$(8)
20 PRINT OCT$(&HF)
30 PRINT OCT$(8^4)
40 PRINT OCT$(8^4-1)
```

```
RUN
10
17
10000
7777
```

PUNTEN OM TE ONTHOUDEN

Als het argument negatief is, wordt de tweecomplementvorm gebruikt, dit is OCT$(-X)=OCT$(65536-X).

FOUTEN

Als er een argument wordt gekozen dat buiten de toegestane waarden valt, volgt een 'Overflow Error'-foutboodschap.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

BIN$
HEX$
Sectie 2, Hoofdstuk 5: Overzicht van getallensystemen die in MSX-BASIC worden gebruikt

ON <uitdrukking> GOSUB

BESCHRIJVING

De ON <uitdrukking> GOSUB <regel1>,<regel2>,<regel3>, ...-opdracht biedt een meerkeuze-mogelijkheid van subroutines. De mogelijkheid hangt af van de meegegeven <uitdrukking>; geeft <uitdrukking> 1 terug, dan wordt de subroutine vanaf regel 1 aangeroepen; wordt 2 teruggegeven, dan regel 2 enz.

De uitdrukking moet een getal opleveren dat varieert van één tot één van de regelnummers die zijn meegegeven.

Bijvoorbeeld:

```
ON X GOSUB 100,250,670,800
```

Als X=1 dan volgt een GOSUB 100.
Als X=2 dan volgt een GOSUB 250.
Als X=3 dan volgt een GOSUB 670.
Als X=4 dan volgt een GOSUB 800.
Als X gelijk is aan nul, of groter dan de meegegeven regelnummers dan zet BASIC de uitvoering voort vanaf de volgende opdracht.

SYNTAX

ON <uitdrukking> GOSUB <lijst met regelnummers>

VOORBEELDEN

```
10 INPUT "1, 2 OF 3";I
20 ON I GOSUB 40,60,80
30 GOTO 10
40 PRINT "SUBROUTINE 1"
50 RETURN
60 PRINT "SUBROUTINE 2"
70 RETURN
80 PRINT "SUBROUTINE 3"
90 RETURN
```

```
RUN
1, 2 OF 3? 2     INVOER 2
SUBROUTINE 2
1, 2 OF 3? 1     INVOER 1
SUBROUTINE 1
1, 2 OF 3? 7     INVOER 7
1, 2 OF 3?       CONTINUEERT VANAF REGEL 30 (GOTO 10)
```

PUNTEN OM TE ONTHOUDEN

Als de geëvalueerde uitdrukking een reëel getal is dan wordt dit getal omgezet naar een integer getal (afgerond), dus X=1.22 wordt dan X=1 en hierna wordt de eerste GOSUB uitgevoerd.

IF
<uitdrukking>=0
OR
<uitdrukking>> aantal items in de lijst AND minder dan 255
THEN
BASIC gaat door met de volgende opdracht

FOUTEN

Levert de uitdrukking een negatief getal op of meer dan 255 dan volgt een 'Illegal Function Call'-foutmelding,

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

ON GOTO
Sectie 1, Hoofdstuk 18: Programmasprongen

ON <uitdrukking> GOTO

BESCHRIJVING
De ON <uitdrukking> GOTO <regel1>,<regel2>,<regel3>, ...-opdracht biedt een meerkeuze-mogelijkheid voor sprongen naar regelnummers. Het gekozen regelnummer hangt af van het resultaat van de <uitdrukking>. Levert de uitdrukking 1 op dan wordt een GOTO <regel1> uitgevoerd; als het resultaat 2 is dan wordt GOTO <regel2> uitgevoerd enz.
De resulterende waarde moet gelijk zijn aan 1 of aan één van de regelnummers die voorkomen in de lijst.
Bijvoorbeeld:

ON X GOTO 100,250,670,800

Als X=1 GOTO regel 100.
Als X=2 GOTO regel 250.
Als X=3 GOTO regel 670.
Als X=4 GOTO regel 800.
Wanneer X=0 of de waarde van X ligt hoger dan één van de regelnummers in de lijst, dan zet BASIC de uitvoering voort vanaf de volgende opdracht.

SYNTAX
ON <uitdrukking> GOTO <lijst met regelnummers>

VOORBEELDEN

```
10 INPUT "HOEVEEL KEER?";N
20 ON N GOTO 60,50,40
30 GOTO 10
40 PRINT "MSX"
50 PRINT "MSX"
60 PRINT "MSX"
```

```
RUN
HOEVEEL KEER?? 2
MSX
MSX
```

PUNTEN OM TE ONTHOUDEN
Wanneer de geëvalueerde waarde een reëel getal oplevert, dan wordt dit omgezet in een integer getal (afgerond), dus als X=1.22 dan wordt dit X=1 en hierna wordt naar het eerste regelnummer uit de lijst gesprongen.

IF

<uitdrukking>=0
OR
<uitdrukking>> aantal items in de lijst AND minder dan 255
THEN
BASIC zet de uitvoering vanaf de volgende opdracht voort.

FOUTEN
Als de uitdrukking een negatief getal oplevert of een waarde groter dan 255, dan volgt een 'Illegal Function Call'-foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ON...GOSUB
Sectie 1, Hoofdstuk 18: Programmasprongen

ON ERROR GOTO

BESCHRIJVING
ON ERROR GOTO activeert de detectie van fouten en geeft aan naar welke regel een GOTO moet worden uitgevoerd zodra de fout optreedt. Zodra de computer weet dat fouten dienen te worden gedetecteerd, springt hij naar de aangegeven regel indien de fout optreedt tijdens de uitvoering van het programma.
Om de foutdetectie te de-activeren moet u een ON ERROR GOTO 0 uitvoeren. Wordt dit gedaan gedurende een foutdetectie-routine dan heeft het tot gevolg dat BASIC wordt onderbroken en de fout die de detectie veroorzaakte wordt afgedrukt.
Met de ERROR-opdracht kunt u uw eigen fouten creëren. Er bestaan ongeveer 36 foutcodes die liggen tussen de foutnummers 1 en 60 in MSX-BASIC. De foutnummers van 60 tot 255 zijn vrij voor gebruik en hierin kunt u uw eigen foutmeldingen creëren.
Om uw eigen fouten te programmeren dient u de computer allereerst in de foutdetectiemodus te plaatsen door aan het begin van uw programma een ON ERROR GOTO <regel> te laten uitvoeren. Als dan in het midden van uw programma, gedurende de uitvoering, een fout optreedt die voldoet aan de door u opgegeven conditie, kunt u uw fout aanroepen met:

IF <conditie> THEN ERROR <foutcode>

ON ERROR GOTO zal worden geactiveerd en de computer zal direct naar de foutafhandelingsroutine springen. U moet een regel binnen de routine hebben met:

IF ERR=<foutcode> THEN PRINT "<foutboodschap>"

De foutdetectiesubroutine kan het programma onderbreken, maar ook met RESUME de uitvoering van het programma voortzetten vanaf de aangegeven regel.

SYNTAX
ON ERROR GOTO <regel>

VOORBEELDEN
Eenvoudige foutdetectie:

```
10 ON ERROR GOTO 1000
20 PRINT "MSX"
30 PRONT "DE FOUT"
40 END
1000 PRINT "FOUT GEDETECTEERD IN REGEL";ERL
1010 END
```

```
RUN
MSX
FOUT GEDETECTEERD IN REGEL 30
Ok
```

PUNTEN OM TE ONTHOUDEN

De MSX-computer zal geen fouten detecteren gedurende de uitvoering van de foutdetectieroutine. Blijkt er een fout te zitten in de foutdetectieroutine dan stopt BASIC en wordt de foutboodschap getoond.

Niet alle fouten kunnen worden behandeld door de foutafhandelingsroutines. Het is daarom aan te bevelen om de foutafhandelingsroutine te beëindigen met ON ERROR GOTO 0, die er voor zorgt dat BASIC stopt en de foutboodschap toont.

Zodra de foutdetectie is geactiveerd, voert de computer een GOTO naar de foutroutine uit, ook al bent u in de commandomodus.

Bent u bezig ue eigen fouten te definiëren, werk dan van 255 naar omlaag.

Ontdekt de computer een fout zonder boodschap dan volgt een "Unprintable Error'-foutmelding.

ON ERROR de-activeert alle evenementbehandelingsroutines, zoals ON INTERVAL en ON STRIG.

FOUTEN

Wanneer u vergeet een regelnummer op te geven dan volgt een 'Undefined Line Number'-foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

ERL
ERR
ERROR
RESUME
Sectie 2, Hoofdstuk 10: Foutafhandeling

ON INTERVAL GOSUB

BESCHRIJVING
ON INTERVAL GOSUB geeft de subroutine aan en de timing van de interruptie in een BASIC-programma. Als de intervalinterruptie actief is (met INTERVAL ON), dan zal de computer na een zeker tijdsverloop een sprong uitvoeren naar een aangegeven subroutine. Dit is één van de evenementafhandelingsopdrachten waarin MSX is gespecialiseerd. (Meer over de toepassing hiervan vindt u in het gedeelte over evenementbehandeling en interruptie door BASIC.)
Het interval wordt berekend in 1/50 seconden en kan iedere willekeurige tijdsduur aannemen.
Zodra een tijdinterruptie optreedt, wordt automatisch een INTERVAL STOP uitgevoerd. Dit voorkomt het optreden van een volgende tijdinterruptie tijdens de uitvoering van de tijdinterruptiesubroutine. Er wordt echter wel onthouden dat er een tijdinterruptie is opgetreden en na voltooiing van de subroutine wordt deze routine opnieuw uitgevoerd. Dit gebeurt niet als de routine zelf de interruptie de-activeert door middel van INTERVAL OFF. Na het voltooien van de interruptiesubroutine zal de computer automatisch een INTERVAL ON uitvoeren, om daarmee de interruptie weer te activeren.

SYNTAX
ON INTERVAL=<tijdinterval in 1/50 van een seconde> GOSUB <regel>

VOORBEELDEN

```
10 ON INTERVAL=500 GOSUB 40
20 INTERVAL ON
30 GOTO 30
40 REM INTERVAL SUBROUTINE
50 PRINT "10 SECONDEN ZIJN VERSTREKEN"
60 PRINT "EINDE PROGRAMMA"
70 END
```

REGEL 10 definieert tijdinterval in 500 eenheden (10 seconden), de subroutine wordt gezet op regel 40.
REGEL 20 activeert de interruptie.

Als u het programma uitvoert krijgt u het volgende resultaat:

```
RUN
10 SECONDEN ZIJN VERSTREKEN
EINDE PROGRAMMA
Ok
```

PUNTEN OM TE ONTHOUDEN
De tijdinterruptie wordt onderbroken zodra BASIC uit een programma raakt: er bestaat dus geen tijdinterruptie in de directe modus.
Ook wordt de tijdinterruptie onderbroken tijdens foutdetectieroutines.

FOUTEN
Vergeet niet de tijdinterruptie aan te schakelen met INTERVAL ON.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
INTERVAL ON/OFF/STOP
Sectie 2, Hoofdstuk 9: Evenementbehandeling en interruptie door BASIC

ON KEY GOSUB

BESCHRIJVING

ON KEY GOSUB definieert de regelnummers voor de functietoetsinterruptie-subroutines. KEY <regelnummer> ON-opdrachten activeren een functietoets-interruptie voor elk van de functietoetsen die kunnen worden gedetecteerd door de computer. Treedt er een interruptie op dan springt de computer naar het corresponderende regelnummer, aangegeven door ON KEY GOSUB.
In de ON KEY GOSUB-opdracht moet u alle regels van subroutines opnemen die bij een functietoets horen. Blijkt er geen subroutine aanwezig te zijn voor een specifieke toets dan wordt dit aangegeven door een komma en wordt het regelnummer overgeslagen.
Zodra een functietoetsinterruptie optreedt, wordt een automatische KEY (<getal>) STOP uitgevoerd. Dit voorkomt dat er een functietoetsinterruptie optreedt tijdens de uitvoering van een functietoetsinterruptiesubroutine. Er wordt echter wel onthouden dat er een interruptie is geweest en de computer zal direct na voltooiing van de routine, dezelfde routine opnieuw uitvoeren. Dit kan voorkomen worden door in de routine zelf de interruptie te laten de-activeren door KEY(<getal>) OFF. Na het verlaten van de interruptiesubroutine voert de computer automatisch een KEY ON uit om daarmee de interruptie te activeren.

SYNTAX

ON KEY GOSUB <lijst met regelnummers>

VOORBEELDEN

Indien u functietoets F1 indrukt zal de computer tweemaal een bieptoon genereren. F2 geeft één keer biep.

```
10 ON KEY GOSUB 50,60
20 KEY(1) ON
30 KEY(2) ON
40 GOTO 40
50 BEEP
60 BEEP
70 RETURN
```

REGEL 10	definieert subroutines voor F1 en F2.
REGEL 20	activeert F1.
REGEL 30	activeert F2.
REGEL 40	oneindige lus die op een functietoets wacht.
REGEL 50	BEEP (F1).
REGEL 60	BEEP (F2 en F1).
REGEL 70	keert terug naar de plaats van herkomst (hier dus regel 40).

PUNTEN OM TE ONTHOUDEN
KEY-interruptie wordt gede-activeerd als het programma niet loopt en ook tijdens de foutdetectieroutines.

FOUTEN
U krijgt een 'Undefined Line Number'-foutboodschap indien u een foutief regelnummer in de lijst plaatst.
Vergeet niet de KEY ON-opdracht te geven.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
KEY ON/OFF/STOP
Sectie 2, Hoofdstuk 9: Evenementbehandeling en interruptie door BASIC

ON SPRITE GOSUB

BESCHRIJVING
ON SPRITE GOSUB activeert de sprite-botsing-interruptie-subroutine. De SPRITE ON-opdracht activeert een detectie die het programma stuurt naar een routine, aangegeven door ON SPRITE GOSUB, zodra twee sprites botsen.
Zodra de interruptie optreedt, wordt een automatische SPRITE STOP uitgevoerd. Dit voorkomt dat een interruptie optreedt tijdens de uitvoering van een interruptiesubroutine.
Er wordt echter wel onthouden dat er sprake is geweest van nog een spritebotsing en de computer zal daarom direct na het beëindigen van de routine, deze routine opnieuw uitvoeren.
Dit wordt voorkomen indien de routine zelf de interruptie de-activeert met een SPRITE OFF. Na het verlaten van de interruptiesubroutine voert de computer automatisch een SPRITE ON uit om de interruptie te activeren.

SYNTAX
ON SPRITE GOSUB <regel>

VOORBEELDEN
In het volgende voorbeeld ziet u twee vierkante sprites, een gele en een witte, die elkaar van de tegenovergestelde zijden van het scherm naderen. Treedt er een botsing op, dan wordt de interruptiesubroutine ge-activeerd en wordt door BASIC een sprong uitgevoerd naar de sprite-interruptiesubroutine. De SPRITE OFF in de sprite-interruptiesubroutine voorkomt verdere detectie van sprite-botsingen. (Probeer deze routine zonder SPRITE OFF en zie wat er gebeurt.)

```
10 ON SPRITE GOSUB 110
20 SCREEN 2,0
30 SPRITE$(0)=STRING$(8,CHR$(255))
40 SPRITE$(1)=STRING$(8,CHR$(255))
50 SPRITE ON
60 FOR I=10 TO 240
70 PUT SPRITE 0,(I,100),11,0
80 PUT SPRITE 1,(250-I,100),15,1
90 NEXT I
100 END
105 REM SPRITE-BOTSINGROUTINE
110 SPRITE OFF
120 BEEP
130 RETURN
```

REGEL 10 definieert de detectieroutine op regel 110.
REGEL 20 zet een grafisch scherm.

REGEL 30	sprite 00 is vierkant.
REGEL 40	ook sprite 1 is vierkant.
REGEL 50	sprite-botsingdetector actief.
REGEL 60	lus.
REGEL 70	sprite (geel) beweegt van links.
REGEL 80	sprite (wit) beweegt van rechts.
REGEL 90	volgende lusdoorgang.
REGEL 100	einde.
REGEL 110	de-activeert de detectie (één keer is voldoende).
REGEL 120	geluid.
REGEL 130	keer terug naar plaats van herkomst.

PUNTEN OM TE ONTHOUDEN

De sprite-interruptie wordt onderbroken zodra het programma niet meer draait, ook tijdens de foutdetectieroutines.

FOUTEN

Geeft u een foutief regelnummer mee voor de interruptiesubroutine dan volgt 'Undefined Line Number'.
Vergeet niet een SPRITE ON-opdracht te geven.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

SPRITE ON/OFF/STOP
SPRITE$
PUT SPRITE
Sectie 2, Hoofdstuk 15: Geavanceerde grafische toepassingen IV: Grafische sprites

ON STOP GOSUB

BESCHRIJVING

ON STOP GOSUB is een interruptie-opdracht, die voorkomt dat de gebruiker een reeds actief programma onderbreekt met <CTRL> <STOP>.

De enige manier om uit een kraakvrij programma te komen, is met behulp van de 'RESET'-knop op de computer. U moet er daarom steeds aan denken dat u uw kraakvrije programma eerst op cassette zet voordat u dat uitvoert.

STOP ON/OFF/STOP activeert/de-activeert STOP KEY-detectie. Wordt STOP ON uitgevoerd dan begint BASIC met de controle van de <CTRL> <STOP> en kijkt of deze wordt ingedrukt na iedere uitgevoerde opdracht. Wordt <CTRL> <STOP> ingetoetst dan wordt BASIC naar de subroutine gestuurd, die wordt aangegeven door de ON STOP GOSUB-opdracht eerder in het programma.

Zodra de interruptie optreedt wordt een automatische STOP STOP uitgevoerd. Dit voorkomt het optreden van een interruptie tijdens de uitvoering van een interruptieroutine. Er wordt echter wel onthouden dat <CTRL> <STOP> werd ingedrukt; na het beëindigen van de subroutine wordt de subroutine direct opnieuw uitgevoerd, tenzij deze routine de interruptie de-activeert. Na het verlaten van de subroutine voert de computer automatisch een STOP ON uit, om daarmee de interruptie te activeren.

SYNTAX

ON STOP GOSUB <regel>

VOORBEELDEN

Dit voorbeeld laat zien hoe de ON STOP GOSUB kan worden gebruikt, om te voorkomen dat een gebruiker in uw programma kijkt. Drukt u <CTRL> <STOP> in dan antwoordt de computer met '<CTRL> <STOP> GEDEACTIVEERD' en gaat door met de uitvoering van het programma. De routine heeft een speciale uitgangsroutine: druk <s> in om te eindigen; anders zal er steeds MSX worden afgedrukt.

```
10 ON STOP GOSUB 100
20 STOP ON
30 IF INKEY$="s" THEN END
40 PRINT "MSX"
50 GOTO 30
90 REM CTRL-STOP SUBROUTINE
100 BEEP
110 PRINT "<CTRL> <STOP> GEDEACTIVEERT"
120 RETURN
```

REGEL 10	STOP subroutine vanaf regel 100.
REGEL 20	activeert de STOP-detectie.
REGEL 30	IF <s> wordt ingedrukt THEN END.

REGEL 40	drukt MSX af.
REGEL 50	een lus met regel 30.
REGEL 100	waarschuwingstoon.
REGEL 110	bericht.
REGEL 120	keert terug naar plaats van herkomst.

PUNTEN OM TE ONTHOUDEN

Let er op dat ON STOP niet voorkomt dat <STOP> het programma laat stilstaan. Alleen het onderbreken van het programma met behulp van <CTRL> <STOP> wordt voorkomen. Probeer <STOP> in te drukken in bovenstaand voorbeeld. U zult zien dat het programma wordt tegengehouden en dat de cursor verschijnt. Drukt u echter de <STOP>-toets voor de tweede keer in, dan gaat het programma gewoon door.

Onthoud dat u STOP ON moet uitvoeren om de STOP-detectie te activeren.

De STOP-interruptie wordt gede-activeerd als het programma niet meer loopt en ook tijdens foutdetectieroutines.

FOUTEN

Wordt een foutief regelnummer meegegeven voor de STOP-subroutine dan volgt 'Undefined Line Number'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

STOP ON/OFF/STOP

Sectie 2, Hoofdstuk 9: Evenementbehandeling en interrupties door BASIC

ON STRIG GOSUB

BESCHRIJVING

ON STRIG GOSUB definieert trekkerinterruptiesubroutines die nodig zijn voor bijvoorbeeld de joysticks en de spatiebalk. STRIG(<n>) ON activeert de trekkerdetectie en wordt samen met ON STRIG GOSUB gebruikt.
Er zijn vijf trekkers:

0	=	spatiebalk
1	=	trekker 1 van joystick 1
2	=	trekker 2 van joystick 2
3	=	trekker 2 van joystick 1
4	=	trekker 2 van joystick 2

In de ON STRIG GOSUB-opdracht moet u de regelnummers meegeven van de subroutines die horen bij elk van de trekkers (behalve wanneer de spatiebalk alleen wordt gebruikt - zie het gedeelte over de syntax). Bestaat er geen subroutine voor een bepaalde trekker dan kunt u deze weglaten en vervangen door een komma.
Zodra de interruptie optreedt, wordt een automatische STRIG(<N>) STOP uitgevoerd ((<N>) is het trekkernummer). Dit voorkomt het optreden van een interruptie tijdens de uitvoering van een interruptieroutine. Er wordt echter wel onthouden dat een interruptie is opgetreden en direct na het beëindigen van de subroutine wordt in dat geval de routine opnieuw uitgevoerd. Dit kan worden voorkomen door de subroutine zelf een STRIG(<n>) OFF te laten uitvoeren. Na het beëindigen van de interruptiesubroutine voert de computer automatisch een STRIG(<n>) ON uit om de interruptie te activeren.

SYNTAX

ON STRIG GOSUB <lijst met regelnummers>
ON STRIG GOSUB <regel> (tenzij de spatiebalk als trekker wordt gebruikt)

VOORBEELDEN

Hier volgt een zeer kort programma dat laat zien hoe de trekkerdetectie werkt, met de spatieblak als trekker. Indien de spatiebalk wordt ingedrukt, springt BASIC naar de trekkerdetectiesubroutine. Anders wordt 'MSX' continu afgedrukt, totdat <s> wordt ingedrukt.

```
10 ON STRIG GOSUB 100
20 STRIG(0) ON
30 IF INKEY$="s" THEN END
40 PRINT "MSX"
50 GOTO 30
90 REM SPATIEBALK TREKKER ROUTINE
100 BEEP
110 PRINT "SPATIEBALK IS INGEDRUKT"
```

```
120 RETURN
```

REGEL 10	activeert de trekkerroutine op regel 100.
REGEL 20	actief.
REGEL 30	na indrukken van <s> eindigen.
REGEL 40	drukt MSX af.
REGEL 50	een lus terug naar regel 30.
REGEL 100	waarschuwingstoon.
REGEL 110	geeft een bericht.
REGEL 120	keert terug naar de plaats van herkomst.

```
RUN
MSX
MSX
MSX
MSX
SPATIEBALK IS INGEDRUKT            druk op de spatiebalk
MSX
MSX
MSX
Ok                                 druk op <s>
```

PUNTEN OM TE ONTHOUDEN]
STRIG wordt gede-activeert als het programma niet meer loopt en ook tijdens de foutdetectieroutines.

FOUTEN
Een veel gemaakte fout:
STRIG (0) ON... Dit is fout en veroorzaakt een 'Syntax Error'. Er mag geen spatie staan tussen STRIG en (0).
U krijgt 'Undefined Line Number' indien u een foutief regelnummer in de lijst opneemt.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
STRING( )
STRIG ON/OFF/STOP
Sectie 2, Hoofdstuk 9: Evenementbehandeling en interrupties door BASIC

OPEN

BESCHRIJVING
OPEN zorgt ervoor dat een aangegeven medium wordt geopend. OPEN reserveert een buffer voor de in- en uitvoer (I/O) en zet de modus voor de I/O-bewerking voor de buffer. Wilt u in- of uitvoer plegen op een bestand dan dient u dit bestand eerst te openen. U moet OPEN hebben uitgevoerd alvorens één van de volgende opdrachten uit te voeren:

PRINT#	PRINT# USING
INPUT#	LINE INPUT#
INPUT$(#)	EOF

Er worden vier soorten randapparatuur door de MSX ondersteund:

CAS:	cassette
CRT:	tekstscherm
GRP:	grafisch scherm
LPT:	regeldrukker of printer

Het aantal randapparaten zal worden uitgebreid bij de introductie van floppy disk-eenheden en andere apparatuur. De beschrijvingen die nodig zijn voor een randapparaat worden meestal aangeleverd in een ROM-cartridge. Vergeet de dubbele punten niet die achter de apparaatbeschrijving staan.
Er bestaan drie verschillende I/O-toestanden:

Output	geeft sequentiële uitvoermodus aan.
Input	geeft sequentiële invoermodus aan.
Append	geeft sequentiële koppelingmodus aan.

<bestandsnummer> is een integere waarde waarvan de waarde kan liggen tussen 1 en MAXFILES, het maximum aantal bestanden.
Bestandsnummers worden ook gebruikt in PRINT#-opdrachten enz. Het <bestandsnummer> is gekoppeld aan een bestand zolang dit niet is gesloten.

SYNTAX
OPEN "<beschrijving randapparaat>[<bestandsnaam>]"[FOR<modus>]AS[#]<bestandsnummer>[<bestandsnaam>],[FOR<modus>]
(, en # zijn niet verplicht)

VOORBEELDEN
Om tekens naar het grafische scherm te schrijven, dient u het bestand te openen naar GRP. Vervolgens kunt u met PRINT# tekens op het scherm afdrukken vanaf de positie van de cursor.

```
10 OPEN "GRP:" FOR OUTPUT AS #1
```

```
20 SCREEN 2
30 DRAW "BM 30,145"
40 PRINT#1,"GRAFISCHE PRINT"
50 GOTO 50
```

REGEL 10	opent bestand naar grafisch scherm.
REGEL 20	grafisch scherm
REGEL 30	verwijst naar het punt 30,145.
REGEL 40	PRINT#1.
REGEL 50	bevriest het grafische scherm.

PUNTEN OM TE ONTHOUDEN

MAXFILES heeft 1 als vervangingswaarde, maar als u naar meer apparaten verwijst in uw programma doet u er goed aan deze waarde te verhogen.

FOUTEN

Is een bestandsnummer hoger dan de door MAXFILES aangegeven waarde dan volgt een 'Bad File Number'-foutboodschap.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

MAXFILES
PRINT#
PRINT# USING
INPUT#
LINE INPUT#
INPUT$
VARPTR
EOF
Sectie 2, Hoofdstuk 14: Geavanceerde grafische toepassingen III: Hoe op een grafisch scherm wordt afgedrukt
Sectie 2, Hoofdstuk 19: Hoe bestanden worden gebruikt

OR Logische OR-bewerker

BESCHRIJVING
OR is één van de logische bewerkers die bitmanipulatie en BOOLEAN algebra uitvoert. Ook wordt OR in de IF...THEN...ELSE-opdracht gebruikt, om meerdere condities te testen voordat resulterende activiteiten worden uitgevoerd.
De waarheidstabel voor OR is als volgt:

OR	X	Y	X OR Y
	0	0	0
	1	0	1
	0	1	1
	1	1	1

Voorbeeld: 34 OR 67

	34	0000000000100010
OR	67	0000000001000011
	99	0000000001100011

OR wordt ook gebruikt in IF...THEN...ELSE-opdrachten en geeft een mogelijkheid tot meerdere condities.

SYNTAX
IF <conditie> OR <conditie> ... THEN ... ELSE
<num var>=<numeriek>OR<numeriek>

VOORBEELDEN
BOOLEAN algebra:

```
10 A=134
20 B=213
30 PRINT "A          ";A;BIN$(A)
40 PRINT "B          ";B;BIN$(B)
50 PRINT "A OR B     ";A OR B;BIN$(A OR B)
```

```
RUN
A  134 10000110
B  213 11010101
A OR B  215  11010111
```

IF <conditie> OR <conditie> THEN:

```
10 INPUT "A=";A
20 INPUT "B=";B
```

```
30 IF A>100 OR B>100 THEN PRINT "OF A OF
B IS GROTER DAN 100"
40 END
```

```
RUN
A=? 78
B=? 107
OF A OF B IS GROTER DAN 100
```

PUNTEN OM TE ONTHOUDEN

U kunt OR gebruiken om twee bytes samen te voegen om zo een bepaalde waarde te creëren.

BOOLEAN-bewerkers moeten een waarde hebben die ligt tussen -32768 en 32767. Reële getallen worden omgezet naar integere waarden.

FOUTEN

Valt de integere waarde buiten de toegestane reeks dan volgt 'Overflow Error'.

Is een operand een string dan treedt een 'Type Mismatch' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

IF
THEN
ELSE
AND
NOT
IMP
XOR
EQV
Sectie 1, Hoofdstuk 6: Het gebruik van condities
Sectie 2, Hoofdstuk 6: BOOLEAN algebra
Sectie 2, Hoofdstuk 7: BOOLEAN II: De IF...THEN...ELSE

OUT

BESCHRIJVING
Deze opdracht verstuurt een byte naar een aangegeven uitvoerpoort. Gebruik deze opdracht nooit in commerciële software, omdat deze zeer machine-afhankelijk is en al snel de MSX-standaard doorkruist.

SYNTAX
OUT <poort>,<gegeven>
<poort>=poortnummer tussen 0 en 255
<gegeven>=gegevensbyte tussen 0 en 255

VOORBEELDEN
```
OUT 1,166
```

PUNTEN OM TE ONTHOUDEN
Gebruik BIOS- in plaats van OUT-opdrachten, opdat u de MSX-standaard handhaaft.

FOUTEN
Ieder getal dat negatief of groter is dan 255 heeft geen betekenis.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
WAIT
INP

PAD touch PAD (aanraakpaneel)

BESCHRIJVING
MSX kan twee 'touch pads' besturen die zijn aangesloten op de joystick-ingangen. PAD geeft de statuswaarde van de 'touch pads'.

SYNTAX
PAD(<n>)
n is 0 tot 3 voor 'touch pads' aangesloten aan joystick-poort 1.

PAD(0) = -1 indien aangeraakt.
= 0 indien niet aangeraakt.
PAD(1) = x-coördinaat (0-255) op pad1.
PAD(2) = y-coördinaat (0-255) op pad1.
PAD(3) = -1 indien 'touch pad'-schakelaar is aangeraakt.
0 indien 'touch pad'-schakelaar niet is aangeraakt.

n is 4 tot 7 voor 'touch pads' aangesloten aan joystick-poort 2.

PAD(4) = -1 indien aangeraakt.
0 indien niet aangeraakt.
PAD(5) = x-coördinaat (0-255) op pad2.
PAD(6) = y-coördinaat (0-255) op pad2.
PAD(7) = -1 indien 'touch pad'-schakelaar is aangeraakt.
0 indien 'touch pad'-schakelaar niet is aangeraakt.

VOORBEELDEN
Hier volgt een kort programma waarbij het 'touch pad' wordt gebruikt als een tekenbord:

```
10 SCREEN 2,0,0
20 IF NOT PAD(0) THEN 20
30 X=PAD(1)
40 Y=INT(PAD(2)*192/255)
50 PSET (X,Y)
60 GOTO 20
```

REGEL 10 HIRES grafische schermmodus.
REGEL 20 indien niets is aangeraakt, dezelfde regel herhalen.
REGEL 30 neem X.
REGEL 40 neem Y, zodat deze op het scherm past.
REGEL 50 drukt een punt af.

PUNTEN OM TE ONTHOUDEN
Let er op dat de coördinaten alleen geldig zijn indien PAD(0) (of PAD(4)) is geëvalueerd. Indien PAD(0) wordt geëvalueerd, worden zowel PAD(5) als

PAD(6) beïnvloed. Indien PAD(4) wordt geëvalueerd, geldt dit ook voor PAD(1) en PAD(2).

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 2, Hoofdstuk 23: Randapparaten

PAINT

BESCHRIJVING
PAINT kleurt een gebied dat is omgeven door een grenslijn van de aangegeven kleur.
U moet de computer laten weten vanaf welke coördinaten de computer met het kleuren moet beginnen. Kleurt u een specifieke vorm, verwijs dan naar willekeurige coördinaten binnen die vorm.

SYNTAX
PAINT <coördinatenspecificatie>
kleurt in de huidige voorgrondkleur.
PAINT <coördinatenspecificatie>,<kleur>
kleurt met de aangegeven kleur.
PAINT <coördinatenspecificatie>,<kleur>,<grenslijnkleur>
kleurt met de aangegeven kleur het gebied omgeven door de grenslijn in de aangegeven kleur (alleen in meerkleurenmodus).

<coördinatenspecificatie>:

1. <x-coördinaat>,<y-coördinaat>
   x-waarde 0-255, y-waarde 0-191.
2. STEP (<x-verplaatsing>,<y-verplaatsing>)
   Deze coördinaten zijn relatief ten opzichte van het door de computer laatst verwezen punt.

Opmerking: <x-verplaatsing>, <y-verplaatsing<, <x-coördinaat> en <y-coördinaat> kunnen zijn: <num var>, <num uitdrukking> of gewoon <num const>.

KLEURCODE

0	transparant	8	middelrood
1	zwart	9	lichtrood
2	middelgroen	10	donkergeel
3	lichtgroen	11	lichtgeel
4	donkerblauw	12	donkergroen
5	lichtblauw	13	paarsrood
6	donkerrood	14	grijs
7	cyaanblauw	15	wit

VOORBEELDEN
1. HIRES grafische modus (SCREEN 2)
In HIRES-modus dient de grenslijnkleur dezelfde kleur te hebben als die voor de inkleuring, omdat anders de kleur overloopt. In dit voorbeeld wordt geen kleur meegegeven, zodat de vervangingskleur wordt gebruikt: dit

betekent dat de lijnen met wit worden gekleurd op een blauwe achtergrond, waarna de vorm wit wordt ingekleurd.

```
10 SCREEN 2
20 PSET (100,100)
30 LINE -STEP(-50,50)
40 LINE -STEP(100,0)
50 LINE -STEP(50,-50)
60 LINE -STEP(-100,0)
70 PAINT (110,110)
100 GOTO 100
```

2. In modus 3, de grafische modus met laag oplossend vermogen, kunt u met een andere kleur inkleuren dan de randkleur.

```
10 SCREEN 3
20 PSET (100,100)
30 LINE -STEP(-50,50)
40 LINE -STEP(100,0)
50 LINE -STEP(50,-50)
60 LINE -STEP(-100,0)
70 PAINT (110,110),9,15
100 GOTO 100
```

PUNTEN OM TE ONTHOUDEN
U kunt de randkleur alleen in meerkleurenmodus opgeven. U kunt slechts één kleur per vorm gebruiken. Dit betekent dat u geen meerkleurenranden kunt hebben.

FOUTEN
Vallen de meegegeven coördinaten buiten het scherm dan volgt een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
DRAW
SCREEN
Sectie 1, Hoofdstuk 27: Kleuren

PDL PADDLE

BESCHRIJVING
PDL geeft de waarde voor PADDLE.
U kunt tot 12 speelpaddles aansluiten, 6 op elk van de joystick-ingangen.

SYNTAX
PDL(<n>)
Poort 1 n=1, 3, 5, 7, 9, 11
Poort 2 n=2, 4, 6, 8, 10, 12
PDL geeft een waarde die ligt tussen 0 en 255.

VOORBEELDEN
```
P1=PDL(1)
```

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 2, Hoofdstuk 23: Randapparatuur

PEEK

BESCHRIJVING
PEEK geeft de waarde van een byte, gelezen vanuit een aangegeven adres in het geheugen.

SYNTAX
PEEK(<integere waarde>)
<integere waarde> moet liggen tussen -32768 en 65535.

VOORBEELDEN

```
10 FOR I=1 TO 10000
20 A=PEEK(I)
30 IF A>31 AND A<127 THEN PRINT
HEX$(I);" ";CHR$(A)
40 NEXT I
```

```
RUN
A &
12 &
19 E
21 j
25 ^
...
```

PUNTEN OM TE ONTHOUDEN
POKE is de tegenovergestelde functie van PEEK. U kunt PEEK niet toepassen op de VIDEO-RAM; gebruik hiervoor VPEEK in de plaats.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
POKE
VPOKE
VPEEK
Zie het overzicht met systeem-variabelen.
Sectie 2, Hoofdstuk 20: Geheugenorganisatie

PLAY

BESCHRIJVING

PLAY speelt muziek die wordt aangegeven met de muziekmacrotaal, die wordt ingevoerd in een stringformaat, zoals dat ook gebeurt met de grafische macrotaal.

De muziekmacrotaal:

A, B, C, D, E, F, G met eventueel #, +, -

Speelt de noot in het huidige octaaf.
of + na de letter geeft een halve toon hoger aan.
- geeft een halve toon lager aan. Let er op dat het gebruik van #, + of - alleen correct is indien een dergelijke toon bestaat. Hij moet dus corresponderen met een zwarte toets op de piano.

O<n> n = 1 t/m 8

Selecteert een octaaf van 1 tot en met 8.
Ieder octaaf loopt van C t/m B.
Het vervangingsoctaaf is 4, waarin C correspondeert met de middelste C op de piano.

N<n> n = 0 t/m 96

Dit laat noot <n> spelen. Dit is een andere manier om iedere noot van elk octaaf te spelen.

L<n> n = 1 t/m 64

Dit bepaalt de duur van de noot.
L1 = HELE NOOT SEMIBREVE
L2 = HALVE NOOT MINIM
L4 = KWARTNOOT CROTCHET
L8 = ACHTSTE NOOT QUAVER
enz.
De duur van een noot wordt aangegeven door <n> te laten volgen op de noot. Bijvoorbeeld L8C is hetzelfde als C8. Dit kan niet worden gedaan met het N-commando. De vervangingsduur is L4.

R<n> n = 1 t/m 64

Rust of pauze. De duur van een rust wordt op eenzelfde manier gezet als bij L.
R1 = HELE NOOT RUST
R2 = HALVE NOOT RUST
R4 = KWARTNOOT RUST
R8 = ACHTSTE NOOT RUST
enz.
Opmerking: R en R0 zijn gelijk aan de vervangingsrust, welke R4 is.

. (stip)

Een stip na een noot A tot G, N of rust R vermeerdert de lengte met de helft van de waarde van die noot, hetzelfde als bij een noot met

een stip. U kunt meerdere stippen achter een noot plaatsen.

T<n> n = 32 t/m 255 TEMPO
Het tempo bepaalt het aantal kwartnoten in een minuut. n mag variëren van 32 t/m 255. Het vervangingstempo is 120.

V<n> n = 0 t/m 15
Volume. V bepaalt het volume van de noten:
0 geeft het laagste volume aan en 15 het hoogste.
Het vervangingsvolume is 8.

M<n> n = 1 t/m 65535
Modulatie. Dit geeft de periode aan van de envelope, aangegeven door S. De vervangingswaarde is hier 255.

S<n> n = 0 t/m 15
Dit bepaalt de vorm van de envelope. Er bestaan verschillende vooraf gedefinieerde envelope-patronen waaruit u een keuze kunt maken. Zie SOUND voor de details.

X<variabele>;
X voert uit wat in de stringvariabele als muziekmacrotaal staat.

In ieder van bovenstaande commando's kan <n> een integere numerieke constante zijn, of een variabele in de vorm van "=<variabele>:" waarin de naam van de variabele is omgeven door '=' en ';'.
Indien u BEEP uitvoert, zet u het geluid van het systeem op de vervangingswaarden.

SYNTAX
PLAY<string-uitdrukking voor stem 1>,<string-uitdrukking voor stem 2>,<string-uitdrukking voor stem 3>
<stem 2 en 3> zijn niet verplicht.

VOORBEELDEN

```
10 A$="CDEFGAB"
20 FOR I=1 TO 8
30 PLAY "O=I;XA$;"
40 NEXT I
```

PUNTEN OM TE ONTHOUDEN
Verwar de functie niet met PLAY( ), die heel anders is.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 28: De muziekmacrotaal

PLAY ()

BESCHRIJVING
Deze functie laat de gebruiker weten dat de computer bezig is met het maken van muziek vanuit een specifiek kanaal (channel).

-1	=	speelt nog
0	=	speelt niet

PLAY(0) controleert alle drie de kanalen en geeft -1 indien één van hen speelt. Speelt geen van hen dan geeft hij 0.

SYNTAX
<num var>=PLAY(<speelkanaal>)
<speelkanaal>=0 t/m 3

PUNTEN OM TE ONTHOUDEN
Deze functie is heel anders dan PLAY.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
PLAY
Sectie 1, Hoofdstuk 28: De muziekmacrotaal

POINT

BESCHRIJVING

POINT geeft de kleur van een specifiek punt. De kleur van een sprite op de voorgrond wordt niet teruggegeven.

SYNTAX

POINT (<x-coördinaat>,<y-coördinaat>)

VOORBEELDEN

Dit voorbeeld is een kort programma dat de kleur van het grafische scherm geeft. Regel 10 geeft de computer aan met PRINT#1 op het grafische scherm af te drukken.

```
10 OPEN "GRP:" FOR OUTPUT AS #1
20 SCREEN 2
30 COLOR 14,8,9
40 CLS
50 P=POINT(100,100)
60 PRINT#1,"KLEUR IS ";P
70 GOTO 70
```

U ziet het scherm rood worden en KLEUR IS 8 wordt afgedrukt.

PUNTEN OM TE ONTHOUDEN

Er wordt 0 teruggegeven in de beide tekstschermen 0 en 1. -1 wordt teruggegeven indien de coördinaten buiten het scherm vallen.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

COLOR
SCREEN
PSET
PRESET
Sectie 1, Hoofdstuk 23: Het afdrukken van punten

POKE

BESCHRIJVING
POKE schrijft een byte naar een aangegeven geheugenadres. POKE wordt hoofdzakelijk gebruikt om machinecode aan een BASIC-programma toe te voegen. Het meest zinvolle gebied voor een machinecoderoutine is het gebruikerswerkgebied, dat kan worden gecreëerd met CLEAR.

SYNTAX
POKE <adres>,<integere uitdrukking>
<adres> kan liggen tussen -32768 en 65535. Is het getal negatief dan kan het adres worden berekend door van 65535 af te trekken.
<integere uitdrukking> moet een 1-bytewaarde zijn, dus deze moet liggen tussen 0 en 255.

VOORBEELDEN
```
POKE 65535,0
```

PUNTEN OM TE ONTHOUDEN
Wees voorzichtig met deze opdracht. U kunt het systeem verstoren indien u op verkeerde plaatsen in het geheugen dingen wijzigt met POKE.
De VIDEO-RAM kunt u wijzigen met VPOKE.

FOUTEN
Vallen <adres> of <integere uitdrukking> buiten de toegestane waarden dan volgt 'Overflow Error'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
CLEAR
PEEK
VPEEK
VPOKE
Zie het overzicht met systeemvariabelen.

POS Positie van de cursor

BESCHRIJVING
POS geeft u de huidige positie van de cursor. Hiervoor heeft u de dummy-variabele (0) nodig.

SYNTAX
POS(0)

VOORBEELDEN
```
LET P=POS(0)
```

PUNTEN OM TE ONTHOUDEN
De meest linksgelegen positie is 0.
De meest rechtsgelegen positie hangt van de schermmodus af.

	WIDTH vervangingswaarde	maximum
scherm 0	37	40
scherm 1	29	32

FOUTEN
Een veel gemaakte fout is het vergeten van de dummy-variabele (0).

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
CSRLIN
WIDTH
Sectie 2, Hoofdstuk 12: Geavanceerde grafische toepassingen I: Karakteristieken van elk der schermen

PRESET

BESCHRIJVING

PRESET zet de kleur van een specifiek punt terug op de kleur van de achtergrond.
Wordt een punt aangegeven dan wordt het punt in deze kleur gewijzigd. (Dit is hetzelfde als PSET.)

SYNTAX

PRESET <coördinatenspecificatie>
PRESET <coördinatenspecificatie>,<kleur>

<coördinatenspecificatie>:

1. (<x-coördinaat>,<y-coördinaat>)
2. STEP (<x-verplaatsing>,<y-verplaatsing>)

<x-verplaatsing>, <y-verplaatsing>, <x-coördinaat> en <y-coördinaat> kunnen <num var>, <num uitdrukking> of gewoon <num const> zijn.

VOORBEELDEN

```
10 SCREEN 2
20 CLS
30 PAINT (10,10),15
40 X=RND(1)*255
50 Y=RND(1)*255
60 PRESET (X,Y)
70 GOTO 40
```

REGEL 10	HIRES grafische modus 2.
REGEL 20	wist het scherm naar blauw.
REGEL 30	kleurt het scherm met wit (voorgrondkleur).
REGEL 40	x-coördinaat.
REGEL 50	y-coördinaat.
REGEL 60	zet de kleur van het punt terug naar de achtergrondkleur (dit is blauw).
REGEL 70	lusbegrenzing.

U ziet een blauw scherm wit worden, waarna verschillende punten op willekeurige plaatsen blauw worden.

PUNTEN OM TE ONTHOUDEN

PRESET <coördinatenspecificatie>,<kleur> is hetzelfde als PSET <coördinatenspecificatie>,<kleur>.

FOUTEN
PRESET doet niets met punten die buiten het scherm vallen. Worden echter coördinaten aangeroepen buiten de toegestane waarden (-32768 t/m 32767) dan volgt een 'Overflow Error'-foutboodschap.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
PSET
POINT
COLOR
Sectie 1, Hoofdstuk 23: Het afdrukken van punten

PRINT

BESCHRIJVING

Deze opdracht is het meest gebruikte commando in BASIC. Het drukt datgene wat u wilt op het scherm af.
Na een PRINT-opdracht kunt u een lijst van gegevens intoetsen die u wilt afdrukken. Dit kunnen numerieke variabelen of stringvariabelen zijn, of strings tussen aanhalingstekens. De afdrukpositie wordt bepaald door de interpunctietekens in de opdracht, of via TAB of LOCATE.
BASIC verdeelt iedere regel in zones, elk 14 tekens lang. Een komma geeft aan dat elk item aan het begin van een 14-tekenzone moet worden afgedrukt.
Een puntkomma geeft aan dat de items achter elkaar op de regel moeten worden afgedrukt. Het intikken van één of meer spaties tussen de items heeft hetzelfde effect als een puntkomma.
Eindigt een PRINT-opdracht op een puntkomma dan drukt de volgende PRINT-opdracht op dezelfde regel af met dezelfde spatiëring als bovengenoemd, dus niet op de volgende regel.
Passen niet alle tekens op een regel dan gaat de computer automatisch op de volgende regel verder.
Afgedrukte getallen worden beëindigd en voorafgegaan door een spatie.
Afgedrukte negatieve getallen worden voorafgegaan door een minteken.
De afkorting voor PRINT is '?'.

SYNTAX

PRINT <lijst met items>
?<lijst met items>

VOORBEELDEN

```
10 PRINT "BOODSCHAPPEN MOETEN TUSSEN
AANHALINGSTEKENS WORDEN GEPLAATST"
20 PRINT "PUNTKOMMA GEEFT AAN DAT DE VOLGENDE
PRINT";
30 PRINT "BEGINT VANAF HET LAATSTE PUNT VAN
DE VOLGENDE PRINT"
40 A=100
50 B=300
60 PRINT "A=";A,"B=";B
70 PRINT "A+B=";A+B
```

```
RUN
BOODSCHAPPEN MOETEN TUSSEN
AANHALINGSTEKENS WORDEN GEPLAATST
PUNTKOMMA GEEFT AAN DAT DE VOLGENDE
PRINT BEGINT VANAF HET LAATSTE PUNT
```

```
VAN DE VORIGE PRINT
A=100    B=300
A+B= 400
Ok
```

Bovenstaand scherm veronderstelt een breedte van 29 tekens.

PUNTEN OM TE ONTHOUDEN
Let er op dat '?' als afkorting automatisch wordt vertaald naar PRINT in de programmalijst.
Het is eenvoudiger om PRINT USING te gebruiken voor het afdrukken van een tabel.

FOUTEN
Gebruik altijd aanhalingstekens voor strings.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
PRINT USING
PRINT#
TAB
LOCATE
Sectie 1, Hoofdstuk 2: Commandomode

PRINT#

BESCHRIJVING
Deze opdracht wordt gebruikt om gegevens te schrijven (af te drukken) op verschillende randapparaten en het grafische scherm. U dient eerst een kanaal (channel) te openen, om vervolgens met het speciale PRINT#-commando te werken. # wordt gevolgd door een bestandsnummer dat wordt aangegeven in de OPEN-opdracht.
PRINT# wordt ook vaak gebruikt om bestanden te bewaren op een cassette. (Zie voorbeeld 2 voor de volledige procedure.)

SYNTAX
PRINT#<bestandsnummer>

VOORBEELDEN

```
10 OPEN "GRP:" FOR OUTPUT AS #1
20 SCREEN 2
30 DRAW "BM30,100"
40 PRINT#1,"PRINT IN HIRES MODUS"
50 GOTO 50
```

REGEL 10	opent een bestand naar het grafische scherm als #1.
REGEL 20	selecteert grafisch scherm 2.
REGEL 30	verplaatst de grafische cursor naar 30,100.
REGEL 40	afdrukken op het grafische scherm.
REGEL 50	bevriest het grafische scherm.

PUNTEN OM TE ONTHOUDEN
Voordat u PRINT# uitvoert, moet er een kanaal (channel) zijn geopend.
INPUT# is het complement van PRINT#.

FOUTEN
Blijkt het aangegeven bestandsnummer niet te zijn geopend, dan volgt een 'Bad File Number'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
MAXFILES
INPUT#
OPEN
CLOSE
Sectie 2, Hoofdstuk 14: Geavanceerde grafische toepassingen III: Hoe op het grafische scherm kan worden afgedrukt met bestanden
Sectie 2, Hoofdstuk 19: Hoe bestanden worden gebruikt

PRINT USING

BESCHRIJVING
De PRINT USING-opdracht stelt u in staat om strings of getallen in een speciaal formaat af te drukken. Het helpt bij het netjes tabuleren van informatie.

SYNTAX
PRINT USING <stringuitdrukking>;<lijst met items>
<lijst met items> kunnen strings of numerieken zijn. Ze moeten worden gescheiden door een puntkomma.
<string-uitdrukking> is één van de speciale formaattekens. Dit kan zowel een string als een stringvariabele zijn.

VERKLARING VAN GEBRUIKTE TEKENS

! geeft aan dat het eerste teken van de string moet worden afgedrukt. F$="MSX":PRINT USING"!";F$

@ voegt de aangegeven string toe aan de positie, aangegeven door @.

```
PRINT USING "ABC@EFG";"MSX"
ABCMSXEFG
```

'#' geeft aan dat een cijfer moet worden afgedrukt, bijv.:

```
PRINT USING "#.###";1.3 drukt af:
1.300
```

dus geeft het formaat aan van het af te drukken getal.
Zoals uit bovenstaand voorbeeld blijkt, kan een decimale punt worden meegegeven in de formaatstring. Heeft het getal minder plaatsen nodig dan is aangegeven met '#', dan wordt het resultaat naar rechts aangepast met behulp van voorafgaande spaties.

```
PRINT USING "###.##";65.87
 65.87
```

\+ en '+' aan het begin of het einde van de formaatstring geeft aan dat het teken van het getal moet worden afgedrukt, dus '+' of '-' op de aangegeven positie.

```
PRINT USING "+###.#####";-0.123422
-0.12342
PRINT USING "####.#+";10.71
  10.7+
```

** de dubbele asterisk geeft aan dat de voorafgaande spaties moeten worden vervangen door '*'. Zij vertegenwoordigen nog twee cijferposities, waar u getallen kunt hebben in het veld:

```
PRINT USING "**#.##";5.55,-5.55
**5.55*-5.55
```

$$ dit plaatst een dollarteken voor het getal (dit is een pondteken in de Engelse tekenset). $$ geeft nog twee cijferposities aan in het veld, waarvan één dus het dollarteken is.

```
PRINT USING "$$#####.##";10000,99999.99,
-100.55
$10000.00$99999.99-$100.55
```

U kunt geen exponentieel formaat gebruiken samen met de $$-aanduiding.

**$ dit teken wordt gecombineerd met ** en $$. Alle blanco posities worden gevuld met * en het teken wordt voorafgegaan door een $. **$ specificeert nog drie cijferposities, waarvan één wordt ingenomen door $.

```
PRINT USING "**$###.##",34.99
***$34.99
```

, een komma die links van de decimale punt in de formaatstring wordt geplaatst, geeft aan dat er een komma moet worden afgedrukt na iedere derde positie, links van de decimale punt.

```
PRINT USING "#######,.##";2000000
2,000,000.00
```

Een komma aan het eind van de formaatstring geeft aan dat de komma aan het einde van het getal moet worden afgedrukt.

```
PRINT USING "##.##,";12.567
12.57,
```

^^^^ dit is de exponentiële formaataanduiding en biedt ruimte voor E+xx. U kunt de positie aangeven van de decimale punt. De significante cijfers worden naar links aangepast en de exponent wordt aangepast.

```
PRINT USING "##.##^^^^";200.00
2.00E+02
```

VOORBEELDEN

```
PRINT USING "$$####,.##";10000;2000.50;3000.555
```

```
$10,000.00    $2,000.50    $3,000.56
```

Er worden veel voorbeelden gegeven in de Geavanceerde Programmeringsgids.

PUNTEN OM TE ONTHOUDEN

Indien het aangegeven getal niet past in het veld dat met de formaattekens wordt aangegeven dan wordt een %-teken voor het getal geplaatst.
Dit gebeurt ook wanneer het afgeronde getal buiten de grootte van het veld terecht komt.
Bijvoorbeeld:

```
PRINT USING "##.##";1000
%1000.00
PRINT USING "##.##";99.999
%100.00
```

FOUTEN

Worden meer dan 24 cijfers aangegeven dan volgt een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

PRINT# USING
Sectie 2, Hoofdstuk 8: PRINT USING

PRINT# USING

BESCHRIJVING
PRINT$ USING is gelijk aan PRINT USING, echter PRINT# USING schrijft naar verschillende randapparaten. U moet een OPEN-opdracht uitvoeren om een kanaal te openen naar een specifiek randapparaat, voordat u deze opdracht gebruikt.

SYNTAX
PRINT#<bestandsnummer>,USING <string-uitdrukking>;<lijst met uitdrukkingen>

VOORBEELDEN
```
PRINT#2,USING"##.###";3.453729
```

PUNTEN OM TE ONTHOUDEN
Er moet een kanaal zijn geopend voordat u PRINT# USING gebruikt.

FOUTEN
Blijkt het opgegeven bestandsnummer niet te zijn geopend dan volgt een 'Bad File Number'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
PRINT USING
MAXFILES
PRINT#
OPEN
CLOSE
Sectie 2, Hoofdstuk 8: PRINT USING
Sectie 2, Hoofdstuk 14: Geavanceerde grafische toepassingen III: Hoe op het grafische scherm wordt afgedrukt met behulp van bestanden
Sectie 2, Hoofdstuk 19: Hoe bestanden worden gebruikt

PSET

BESCHRIJVING
PSET drukt een punt af op de aangegeven coördinaten in de huidige voorgrondkleur of de aangegeven kleur.

SYNTAX
PSET <coördinatenspecificatie>
PSET <coördinatenspecificatie>,<kleur>

<coördinatenspecificatie>:
1. (x-coördinaat>,<y-coördinaat>)
2. STEP (<x-verplaatsing>,<y-verplaatsing>)

<x-verplaatsing>, <y-verplaatsing>, <x-coördinaat>, <y-coördinaat> en <kleur> kunnen <num var>, <num uitdrukking> of gewoon <num const> zijn.

VOORBEELDEN

```
10 COLOR 15,1,1
20 SCREEN 2
30 X=RND(1)*255
40 Y=RND(1)*255
50 Z=INT(RND(1)*16)
60 PSET (X,Y),Z
70 GOTO 30
```

REGEL 10 zwarte rand en achtergrond.
REGEL 50 willekeurige kleur.

PUNTEN OM TE ONTHOUDEN
PRESET met een aangegeven kleur is gelijk aan PSET met aangegeven kleur.

FOUTEN
PSET doet niets met punten die buiten het scherm vasllen. Wordt een integer gebruikt, die buiten de toegestane waarden valt dan volgt een 'Overflow Error'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
COLOR
PRESET
Sectie 1, Hoofdstuk 23: Het afdrukken van punten

PUT SPRITE

BESCHRIJVING

Deze opdracht plaatst een sprite op het scherm. U kunt één sprite per sprite-vlak plaatsen. Er zijn 32 sprite-vlakken, daarom is het maximale aantal sprites dat tegelijkertijd kan worden afgebeeld 32.
De coördinatenspecificatie vertelt de computer waar de sprite moet worden geplaatst. U kunt een sprite verbergen voorbij de rand van het beeldscherm, maar u kunt een sprite ook voorlangs of achterlangs een andere sprite bewegen.
U kunt elk van de 16 kleuren gebruiken. U kunt echter maar één kleur per sprite gebruiken.
Zie het gedeelte over sprite in de Geavanceerde Programmeergids, met betrekking tot het gebruik van sprites.

SYNTAX

PUT SPRITE<sprite-vlaknummer>[,<coördinatenspecificatie>][,<kleur>][,<patroonnummer>]
Ieder van de argumenten kan zijn: <num const>, <num var> of <num uitdrukking>, maar dan wel met toegestane waarden.
<sprite-vlaknummer>=0 t/m 31.
<coördinatenspcificatie>:

1. (<x-coördinaat>,<y-coördinaat>)
2. STEP (<x-verplaatsing>,<y-verplaatsing>)

Coördinatenwaarden van -32768 t/m 32767.

Sprite-schermcoördinaten als volgt: X=-32 t/m 255
Y=-32 t/m 191

Vallen de coördinaten buiten het scherm dan treedt het zogenaamde 'wrap around'-mechanisme op (zie de Geavanceerde Programmeergids).

<patroonnummer> = 0 t/m 256 bij sprites van 8x8 pixels.
= 0 t/m 64 bij sprite van 16x16 pixels.

VOORBEELDEN

```
10 SCREEN 2
20 SPRITE$(0)=STRING$(8,CHR$(255))
30 X=100:Y=120
40 C=7
50 PUT SPRITE 0,(X,Y),C,0
60 GOTO 60
```

REGEL 10	selecteert schermmodus 2.
REGEL 20	definieert een vierkante sprite (0).
REGEL 30	X- en Y-coördinaten.
REGEL 40	kleur is cyaan (7).
REGEL 50	plaatst sprite-patroon 0 in sprite-vlak 0, bij de coördinaten X,Y in cyaan.

PUNTEN OM TE ONTHOUDEN

Indien 208 (&HD0) of 209 (&HD1) wordt toegekend aan de Y-coördinaat dan heeft dit tot resultaat dat één of meer sprites tijdelijk worden verborgen. +(Zie de Geavanceerde Programmeergids: Het gedeelte over de toepassing van sprites.)

U kunt in tekstmodus 0 geen sprites gebruiken.

De sprite-grootte wordt bepaald via de SCREEN-opdracht.

Wordt de <coördinatenspecificatie> weggelaten dan plaatst de computer de sprite bij het laatst verwezen punt op het beeldscherm.

FOUTEN

Valt de <coördinatenspecificatie> buiten de toegestane waarden dan treedt een 'Overflow Error' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

ON SPRITE GOSUB
SPRITE$
SPRITE ON/OFF/STOP
SCREEN
COLOR
Sectie 1, Hoofdstuk 15: Geavanceerde grafische toepassingen IV: Grafische sprites

READ

BESCHRIJVING
Deze opdracht leest gegevens uit DATA-regels en kent deze gegevens toe aan variabelen die in de READ-opdracht staan.
READ wordt altijd samen met DATA gebruikt. De gegevens staan altijd opgeslagen in de DATA-regels, deze regels kunnen door de computer niet worden uitgevoerd. Om toegang te verkrijgen tot de DATA-gegevens dient een READ te worden gebruikt.
U kunt een lijst met variabelen inlezen in een READ-opdracht. De variabelen kunnen worden ingelezen in arrays, stringvariabelen of numerieke variabelen, zolang de variabelensoorten maar overeenstemmen.
READ leest vanaf het laagste regelnummer naar het hoogste regelnummer. Wilt u echter vanaf een specifieke regel met lezen beginnen dan kunt u dit aangeven met de RESTORE-opdracht; deze verwijst naar de DATA-regel waar READ dient aan te vangen.

SYNTAX
READ <variabelen>

VOORBEELDEN

```
10 READ A$,B
20 PRINT A$;B
30 READ C$,D
40 PRINT C$,D
50 DATA "ASTRONOMIE",500,"NATUURKUNDE",600
```

```
RUN
ASTRONOMIE 500
NATUURKUNDE 600
```

PUNTEN OM TE ONTHOUDEN
Zoals blijkt uit bovenstaand programma, onthoudt de computer het laatste DATA-element. Iedere volgende READ-opdracht leest vanaf het eerste ongelezen element.

FOUTEN
Stemmen de variabelensoorten niet overeen dan volgt een 'Syntax Error'.
Wordt er een READ uitgevoerd zonder dat er nog DATA-elementen aanwezig zijn, dan volgt een 'Out Of Data'-foutboodschap; denk eraan RESTORE te gebruiken.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
DATA
RESTORE
Sectie 1, Hoofdstuk 12: Het lezen van gegevens in arrays

REM

BESCHRIJVING
REM biedt de programmeur een mogelijkheid om commentaarregels in zijn programma op te nemen. Een REM-regel wordt niet uitgevoerd, dus de computer slaat deze regel over zodra hij hem tegenkomt.
REM-regels worden gebruikt om de programma's te documenteren. Tijdens het schrijven van programma's doet u er goed aan bepaalde, bij elkaar horende regels, te voorzien van commentaar. U vergeet anders snel wat u met bepaalde programmagedeelten wilde doen. Door goed te documenteren met REM helpt u uzelf te herinneren hoe een programma werkt. REM-regels maken uw programma's beter te begrijpen voor andere mensen.
REM-regels worden ook als titel van een programma gebruikt.
REM-regels kunnen worden afgekort met een apostrof (').

SYNTAX
REM commentaar ...
' commentaar ...

VOORBEELDEN

```
10 REM MIJN EERSTE PROGRAMMA
20 REM TITEL : PROEFPROG
30 REM DATUM : 3 / 9 / 1985
40 REM VERSIE: 1
50 REM
...
...
rest van het programma
```

PUNTEN OM TE ONTHOUDEN
REM kan ook worden toegevoegd aan het einde van een regel (:REM...).
Doe dit echter niet bij DATA-regels, omdat de computer veronderstelt dat dit extra gegevens zijn.
REM-opdrachten maken het programma beter begrijpbaar, maar kosten kostbare geheugenruimte.
U kunt REM-regels toepassen zolang er voldoende geheugenruimte aanwezig is, totdat er een geheugentekort blijkt te zijn. Verwijder dan de REM-regels.

FOUTEN
Voer nooit een GOTO of een GOSUB naar een REM-regel uit. Verwijdert u een REM-opdracht, welke wordt gerefereerd vanuit een GOTO of een GOSUB, dan volgt een 'Undefined Number Error'-foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 3: Het schrijven van een programma

RENUM

BESCHRIJVING
Deze opdracht nummert het programma opnieuw. Het gebeurt wel eens dat uw programma er chaotisch uitziet omdat de regelnummers niet in de vaste verhoging staan. Gebruik RENUM: dit is de beste oplossing voor het verbeteren van het aanzien van een programmalijst en daarbij komt nog dat het toevoegen van programmaregels eenvoudiger wordt.
RENUM op zichzelf nummert opnieuw vanaf regel 10, met een verhogingsfactor van 10; dit is de meest toegepaste regelnummering, maar u kunt vanaf een willekeurige regel met hernummeren beginnen, met elke willekeurige verhoging.
RENUM zal automatisch de regelnummers aanpassen van GOTO, GOSUB, RESTORE, THEN, ELSE, ON GOTO en ON GOSUB.

SYNTAX
RENUM
Nummert vanaf regel 10 met verhoging 10.
RENUM <regel1>
Nummert vanaf <regel1> met verhoging 10.
RENUM <regel1>,<regel2>
Nummert vanaf <regel1> tot <regel2> met verhoging 10.
RENUM <regel1>,<regel2>,<verhoging>
Nummert vanaf <regel1> tot <regel2> met de opgegeven verhoging.
RENUM <regel2>,<verhoging>
Nummert vanaf oud regelnummer 2, met nieuw regelnummer 1 en gegeven verhoging.
RENUM <regel2>
Nummert vanaf oud regelnummer 2, met nieuw regelnummer 10 en verhoging 10.
RENUM,,<verhoging>
Nummert opnieuw vanaf oud regelnummer 10 met nieuw regelnummer 10 en de gegeven verhoging.
RENUM <regel1>,,<verhoging>
Nummert vanaf oud regelnummer 10, met nieuw regelnummer 1 en de gegeven verhoging.

VOORBEELDEN
```
RENUM 1000,200,20
```
Nummert vanaf oud regelnummer 200, te beginnen met nieuw regelnummer 1000 en verhoging 20.

```
10 REM
20 REM
30 REM
40 REM
50 REM
```

```
RENUM ,10,20
Ok
LIST
10 REM
30 REM
50 REM
70 REM
90 REM
```

FOUTEN

Wordt er vanuit GOTO of GOSUB naar een niet-bestaand regelnummer verwezen (of vanuit een andere opdracht met een regelnummer), dan volgt een 'Undefined Line NNN in MMMM'-foutboodschap.
Bijvoorbeeld:

```
10 GOTO 97
RENUM
Undefined Line 97 in 10
```

Zonder meer opnieuw nummeren, zoals RENUM 10,40 met oude regelnummers als 10, 20, 30 enz. werkt niet. Er volgt dan een 'Illegal Function Call'.
Negatieve verhogingen zijn niet toegestaan.
U kunt geen regelnummers maken die groter zijn dan 65535. Wanneer blijkt dat gedurende het hernummeren een regelnummer optreedt dat hoger is dan deze waarde dan treedt een 'Illegal Function Call' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's.
Sectie 2, Hoofdstuk 1: Geavanceerde programmabewerking.

RESTORE

BESCHRIJVING

RESTORE van DATA-gegevens. Wanneer de computer DATA inleest met READ-opdrachten, wordt daarmee begonnen vanaf de DATA-regel met het laagste regelnummer, in oplopende regelnummers totdat geen DATA-elementen meer worden gevonden. Om de computer vanaf een specifieke regel te laten lezen met READ kunt u RESTORE uitvoeren met een verwijzing naar de DATA-regel vanwaar met lezen dient te worden begonnen.

Na RESTORE leest READ vanaf de DATA-regel die is aangegeven in de READ-opdracht. Blijkt geen regelnummer te zijn opgegeven dan wordt gelezen vanaf het eerste DATA-element dat de computer tegenkomt.

SYNTAX

RESTORE
RESTORE <regel>

VOORBEELDEN

```
10 READ A$
20 PRINT A$
30 RESTORE
40 READ B$
50 READ B$
60 DATA "GOUDEN JAREN"
```

```
RUN
GOUDEN JAREN
GOUDEN JAREN
Ok
```

PUNTEN OM TE ONTHOUDEN

Gebruik RESTORE om 'Out of Data'-foutmeldingen te voorkomen.

FOUTEN

Verwijst RESTORE naar een niet-bestaand regelnummer dan volgt 'Undefined Line Number'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

READ
DATA
Sectie 1, Hoofdstuk 12: Het lezen van gegevens in arrays

RESUME

BESCHRIJVING

RESUME geeft aan dat de computer met de uitvoering dient verder te gaan na een foutdetectieroutine, veroorzaakt door een foutdetectie-opdracht ON ERROR GOTO. Nadat de fout is afgegeven, geeft RESUME aan vanwaar in het programma dient te worden verder gegaan.
Is er geen RESUME aanwezig in een foutdetectieroutine dan volgt een 'NO RESUME'-foutmelding, tenzij het programma wordt tegengehouden met STOP of END in de foutroutine.
Er zijn drie vormen van RESUME.

SYNTAX

RESUME of RESUME 0
Gaat met de uitvoering verder vanaf de opdracht die de fout veroorzaakte.
RESUME NEXT
Gaat met de uitvoering verder vanaf de opdracht die volgt op de opdracht die de fout veroorzaakte.
RESUME <regel>
Gaat verder met de uitvoering vanaf het aangegeven regelnummer.

VOORBEELDEN

In deze routine worden alle opdrachten die worden ingevoerd en de string 'DOODT' bevatten, behandeld als een nieuwe fout (nr. 255) en door de foutroutine afgehandeld via de ERR-functie. RESUME 20 geeft aan dat na de fout moet worden doorgegaan vanaf regel 20.

```
10 ON ERROR GOTO 1000
20 INPUT "MAJESTEIT, WAT IS UW BEVEL ";A$
30 IF INSTR(A$,"DOODT") THEN ERROR 255
40 PRINT "OK.":END
...
...
1000 IF ERR=255 THEN PRINT "DODEN IS NIET TOEGESTAAN
IN DIT AVONTUUR.":RESUME 20
1010 END
```

```
RUN
MAJESTEIT, WAT IS UW BEVEL? DOODT HEM
DODEN IS NIET TOEGESTAAN IN DIT AVONTUUR.
MAJESTEIT, WAT IS UW BEVEL? GA NAAR OOST
OK.
```

PUNTEN OM TE ONTHOUDEN

U kunt RESUME niet gebruiken om vanuit de commandomodus in BASIC te komen.

FOUTEN
Een ontbrekende RESUME-opdracht in een foutroutine veroorzaakt een 'RESUME without'-foutmelding.
Wordt vanuit RESUME naar een niet bestaand regelnummer verwezen dan volgt een 'Undefined Line Number'-foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ON ERROR GOTO
ERL
ERR
ERROR
Sectie 2, Hoofdstuk 10: Foutafhandeling

RETURN

BESCHRIJVING
RETURN zorgt voor een terugkeer vanuit een GOSUB-subroutine. Deze opdracht zorgt er voor dat de computer terugspringt naar de opdracht die volgt op de meest recente GOSUB-opdracht (die verantwoordelijk was voor de uitvoering van de subroutine).

SYNTAX
RETURN

VOORBEELDEN

```
10 PRINT 1000
20 GOSUB 50
30 PRINT 2000
40 STOP
50 PRINT "SPRING NAAR DE SUBROUTINE"
60 RETURN
```

```
RUN
1000
SPRING NAAR DE SUBROUTINE
2000
BREAK IN 40
```

PUNTEN OM TE ONTHOUDEN
Het is zeer eenvoudig om stuk te lopen in een subroutine. U doet er daarom goed aan de subroutines goed van elkaar te scheiden met END of STOP aan het einde van het hoofdprogramma, om ongewenste aanroep van de subroutine te voorkomen.
Indien de logica van uw programma het toestaat, is het heel wel mogelijk om meerdere RETURN-opdrachten in één subroutine te plaatsen.
RETURN wordt ook gebruikt in verschillende evenementbehandeling-subroutines.

FOUTEN
Een RETURN zonder GOSUB veroorzaakt een 'RETURN without GOSUB'-foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
GOSUB

ON...GOSUB
ON ERROR GOSUB
ON INTERVAL GOSUB
ON KEY GOSUB
ON SPRITE GOSUB
ON STOP GOSUB
ON STRIG GOSUB
Sectie 1, Hoofdstuk 17: Gestructureerd programmeren

RIGHT$

BESCHRIJVING
Dit is één van de stringmanipulatie-functies. RIGHT$(A$,n) geeft de meest rechtsgelegen n tekens in een string A$. Is n gelijk aan de lengte van A$ dan is het duidelijk dat u de gehele string krijgt. Is n gelijk aan 0 dan wordt de nulstring teruggegeven.

SYNTAX
RIGHT$(<string-uitdrukking>,<integere numerieke waarde>)

VOORBEELDEN

```
10 A$="HOOFDBASIS AAN TOM, OVER"
20 PRINT RIGHT$(A$,4)
30 B$="AFTELLEN BEGINT, MOTOREN AAN"
40 C=11
50 D$=RIGHT$(B$,C)
60 PRINT D$
```

```
RUN
OVER
MOTOREN AAN
```

FOUTEN
Het argument van RIGHT$ is een string, gevolgd door een getal. Staan deze in de verkeerde volgorde dan krijgt u een 'Type Mismatch Error'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
LEFT$
MID$
INSTR
Sectie 1, Hoofdstuk 14: Het manipuleren van strings

RND

BESCHRIJVING
De RND-functie genereert een willekeurig getal met een waarde die ligt tussen 0 en 1. Steeds dezelfde getallen worden gegenereerd zodra het programma weer wordt uitgevoerd met RUN. Om echte willekeurige getallen te verkrijgen die deze procedure niet volgen, kunt u RND(-TIME) gebruiken.

SYNTAX
RND(<numeriek>)
Indien <numeriek> negatief is dan volgt een verschillende reeks van getallen volgens de aangegeven waarde.
Is <numeriek> gelijk aan 0 dan wordt het getal herhaald.
Indien <numeriek> positief is, wordt de volgende reeks willekeurige getallen gegenereerd.

VOORBEELDEN

```
10 FOR I=1 TO 5
20 PRINT RND(1)
30 NEXT I
```

```
RUN
.59521943994623
.10658628050158
.76597651772822
.57756392935958
.73474759503023
Ok
```

Opmerking: Dit programma genereert na iedere RUN dezelfde serie willekeurige getallen.
Om een willekeurig getal te genereren tussen 0 en 9, kunt u de volgende functie gebruiken.

```
X=INT(RND(1)*10)
```

PUNTEN OM TE ONTHOUDEN
RND genereert 14 cijfers met dubbele nauwkeurigheid, tussen 0 en 0.99999999999999.
RND(-TIME) geeft werkelijk willekeurige getallen, terwijl RND(1) eenzelfde reeks willekeurige getallen geeft.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 15: Functies

RUN

BESCHRIJVING

RUN voert een programma uit.

SYNTAX

RUN

RUN <regel> ... voert een programma uit vanaf de gegeven regel.

VOORBEELDEN

```
10 PRINT 10000
20 A=300
30 B=200
40 PRINT A+B
```

```
RUN
10000
500
Ok

RUN 20
500
Ok
```

PUNTEN OM TE ONTHOUDEN

END of STOP in een programma beëindigen de uitvoering.
<CTRL> <STOP> stopt de uitvoering vanaf het toetsenbord. Na het aanschakelen van de computer is functietoets F4 geprogrammeerd als RUN <wagen terug>.

FOUTEN

Indien er geen BASIC-programma in het geheugen aanwezig is, levert RUN een 'Ok'-bericht op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

END

STOP

Sectie 1, Hoofdstuk 3: Het schrijven van een programma

Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's

SAVE

BESCHRIJVING
Deze opdracht schrijft een BASIC-programma in de vorm van een ASCII-bestand naar een opgegeven medium.
SAVE wordt gebruikt bij het samenvoegen van programma's. Zie hiervoor MERGE en het gedeelte over de cassettebewerkingen in de Geavanceerde Programmeergids.

SYNTAX
SAVE "<programmanaam>"
SAVE "<medium>[<bestandsnaam>]"
<medium> = CAS: voor een cassette. Andere randapparatuur zou bijvoorbeeld de floppy disk kunnen zijn. In de meeste huidige versies van MSX-BASIC worden echter nog geen andere randapparaten gevoerd.

VOORBEELDEN
Veronderstel het volgende programma in het geheugen van de computer:

```
10 PRINT "HALLO"
20 PRINT "WELKOM"
30 PRINT "BIJ UW"
50 FOR I=1 TO 10
60 PRINT "MSX"
70 NEXT I
```

Om dit te bewaren in een ASCII-bestand met de naam 'PROG', kunt u het volgende doen:

SAVE "PROG" of SAVE "CAS:PROG"

Om het programma vervolgens weer te laden:

LOAD "PROG" of LOAD "CAS:PROG"

PUNTEN OM TE ONTHOUDEN
SAVE is anders dan CSAVE omdat SAVE gegevens in een ASCII-bestand opslaat, terwijl CSAVE in een afkortingen-formaat wordt opgeslagen.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
MERGE
LOAD
Sectie 2, Hoofdstuk 11: Het bewaren en laden via de cassetterecorder

SCREEN

BESCHRIJVING

SCREEN heeft een aantal functies. Deze opdracht wordt gebruikt voor de verschillende modi van uitvoering voor het beeldscherm, de cassette-recorder, het toetsenbord, de sprites en de afdrukeenheid. SCREEN wordt echter hoofdzakelijk gebruikt voor het zetten van de beeldschermmodus.
Met de SCREEN-opdracht kunt u de sprite-grootte meegeven, maar u kunt slechts één grootte per keer opgeven.
Andere mogelijkheden voor SCREEN zijn: het zetten van het toetsaanslag-signaal (de 'KLICK'), het bepalen van de BAUD-waarde voor cassette en de printerselectie.

SYNTAX

SCREEN[<modus>][,<sprite-grootte>][,<toetssignaal>][,<cassette-BAUD-waarde>][,<printeroptie>]

	VOORBEELD
<modus>=0, 1, 2, 3	
0=40x24 tekstmodus	SCREEN 0
1=32x24 tekstmodus	SCREEN 1
2=HIRES-modus	SCREEN 2
3=meerkleurenmodus	SCREEN 3
<sprite-grootte>=0, 1, 2, 3	
0=8x8 onvergroot	SCREEN,0
1=8x8 vergroot	SCREEN,1
2=16x16 onvergroot	SCREEN,2
3=16x16 vergroot	SCREEN,3

Opmerking: wordt de sprite-grootte meegegeven dan wordt de inhoud van SPRITE$ gewist.

<toetssignaal>=0 of niet-nul		
0	= de-activeer de 'klick'	SCREEN,,0
niet 0	= activeer de 'klick'	SCREEN,,1
<cassette BAUD-waarde>=0, 1		
0 =	1200 BAUD	SCREEN,,,0
1 =	2400 BAUD	SCREEN,,,1

Opmerking: de BAUD-waarde kan worden gewijzigd met CSAVE.

<printeroptie>=0 of niet 0		
0	= MSX-printer met de MSX-grafische mogelijkheden	SCREEN,,,,0
niet 0	= niet-MSX-printer en alle grafische tekens worden spaties	SCREEN,,,,1

VOORBEELDEN
Dit voorbeeld laat zien waarin de grafische schermen in modus 2 (HIRES) en 3 (LOWRES) verschillen. Het programma voert dezelfde taak uit in iedere modus, waaruit dan de voordelen en de nadelen van elk der schermen blijkt.

```
10 SCREEN 2
20 GOSUB 90
30 PAINT (105,120)
40 FOR I=1 TO 1000:NEXT
50 SCREEN 3
60 GOSUB 90
70 PAINT (105,120),8,15
80 GOTO 80
90 PSET (100,100)
100 DRAW "R50D50L50U50F50"
110 RETURN
```

REGEL 10	HIRES grafische modus.
REGEL 20	spring naar de subroutine.
REGEL 30	inkleuren met de huidige voorgrondkleur.
REGEL 40	vertraging.
REGEL 50	meerkleuren grafische modus.
REGEL 60	spring naar de subroutine.
REGEL 70	inkleuren met rood, van het gebied dat wordt begrensd door de lijn.
REGEL 90	subroutine: positioneert de grafische cursor.
REGEL 100	tekent een vierkant met een diagonaal.
REGEL 110	keer terug naar de hoofdroutine.

PUNTEN OM TE ONTHOUDEN
Karakteristieken van elk der schermen worden gegeven in het gedeelte over geavanceerde grafische toepassingen.

FOUTEN
Het gebruik van INPUT in schermmodus 2 of 3 brengt u terug in de laatst gebruikte tekstmodus.
Het gebruik van een grafische opdracht in de tekstmodus, behalve PUT SPRITE in modus 1, resulteert in een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 8: Meer over PRINT en het beeldscherm
Sectie 1, Hoofdstuk 21: De verschillende schermen
Sectie 2, Hoofdstuk 11: Het bewaren en laden via de cassetterecorder

Sectie 2, Hoofdstuk 12: Geavanceerde grafische toepassingen I: Karakteristieken van elk scherm
Sectie 2, Hoofdstuk 15: Geavanceerde grafische toepassingen II: Grafische sprites
Sectie 2, Hoofdstuk 23: Randapparatuur

SGN

BESCHRIJVING
SGN geeft het teken van een gegeven getal.
-1 indien het argument negatief is.
0 indien het argument nul is.
1 indien het argument positief is.

SYNTAX
SGN(<numeriek>)

VOORBEELDEN
```
PRINT SGN(-1000)
-1
```

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ABS
Sectie 1, Hoofdstuk 15: Functies

SIN SINus

BESCHRIJVING
SIN geeft de sinus van het argument in radialen.

SYNTAX
SIN(<numeriek>)

VOORBEELDEN
```
PRINT SIN(1)
.84147098480792
```

PUNTEN OM TE ONTHOUDEN
SIN berekent met dubbele nauwkeurigheid en geeft tot 14 significante cijfers.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
COS
TAN
ATN
Sectie 1, Hoofdstuk 19: Wiskundige functies

SOUND

BESCHRIJVING
SOUND schrijft direct naar de programmeerbare 'SOUND GENERATOR CHIP'. De PSG heeft 14 geluidsregisters. Voor een nauwkeurige uitleg van de SOUND-opdracht kunt u het gedeelte over geavanceerde geluidseffecten raadplegen.

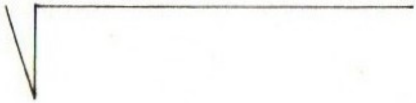
REGISTER	REEKS	BESCHRIJVING
0	0-255	frequentiekanaal A (fijn)
1	0-15	frequentiekanaal A (grof)
2	0-255	frequentiekanaal B (fijn)
3	0-15	frequentiekanaal B (grof)
4	0-255	frequentiekanaal C (fijn)
5	0-15	frequentiekanaal C (grof)
6	0-31	frequentieruis
7	0-63	mixage: toon en ruis. Iedere bit heeft een eigen besturing:
	bit 0	toonkanaal A, 0=ON 1=OFF
	bit 1	toonkanaal B, 0=ON 1=OFF
	bit 2	toonkanaal C, 0=ON 1=OFF
	bit 3	ruiskanaal A, 0=ON 1=OFF
	bit 4	ruiskanaal B, 0=ON 1=OFF
	bit 5	ruiskanaal C, 0=ON 1=OFF
8	0-16	volumekanaal A (envelope met 16)
9	0-16	volumekanaal B (envelope met 16)
10	0-16	volumekanaal C (envelope met 16)
11	0-255	envelope-periode (hoog)
12	0-255	envelope-periode (hoog)
	Envelope-periode (0-65535) = R12*226+R11.	

REGISTER	ENVELOPE NR	TOONVORM
13	0, 1, 2, 3, 9	
	4, 5, 6, 7, 15	
	8	

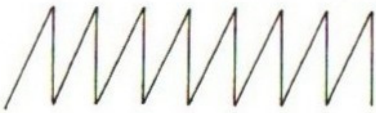
10



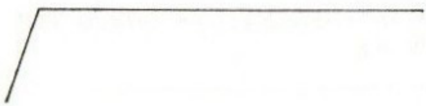
11



12



13



14



SYNTAX
SOUND <register>,<gegevens>

VOORBEELDEN
Speciaal geluidseffect:

```
10 SOUND 7,62
20 SOUND 1,0
30 SOUND 0,254
40 SOUND 8,16
50 SOUND 13,9
60 SOUND 12,60
70 SOUND 11,0
```

FOUTEN
Blijkt na een SOUND-opdracht dat u niets te horen krijgt, controleer dan eerst de syntax en vervolgens of het volume van uw TV correct is ingesteld.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
PLAY
Sectie 2, Hoofdstuk 19: Geavanceerde geluidseffecten met de PSG

SPACE$

BESCHRIJVING
SPACE$ geeft een string met een gegeven aantal spaties. Het argument moet reëel zijn en liggen tussen 0 en 255. Gedeelten van reële getallen worden genegeerd.

SYNTAX
SPACE$(<numeriek>)
<numeriek> = 0 - 255

VOORBEELDEN

```
10 A$="MSX"+SPACE$(5)+"COMPUTER"
20 PRINT A$
```

```
RUN
MSX     COMPUTER
```

PUNTEN OM TE ONTHOUDEN
Er bestaat een soortgelijke functie SPC, die ook blanco spaties afdrukt, maar deze functie kan alleen worden gebruikt in PRINT- en LPRINT-opdrachten.

FOUTEN
Er treedt een 'Type Mismatch'-foutmelding op wanneer SPACE$ aan een numerieke variabele wordt gelijkgesteld.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
SPC
Sectie 1, Hoofdstuk 9: Meer over PRINT en het beeldscherm

SPC

BESCHRIJVING
SPC drukt spaties af op het scherm. SPC kan alleen samen worden gebruikt met PRINT en LPRINT. Deze functie wordt hoofdzakelijk gebruikt om geheugenruimte uit te sparen, indien u veel spaties nodig heeft in een PRINT-opdracht.

SYNTAX
SPC(<numeriek>)
<numeriek>=0 - 255

VOORBEELDEN

```
10 A$="MSX":B$="COMPUTER"
20 PRINT A$;SPC(10);B$
```

```
RUN
MSX          COMPUTER
```

PUNTEN OM TE ONTHOUDEN
SPC is iets anders dan SPACE$, omdat er geen sprake is van een string.

FOUTEN
Is het argument groter dan 255 dan volgt een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
PRINT
LPRINT
SPACE$
Sectie 1, Hoofdstuk 9: Meer over PRINT en het beeldscherm

SPRITE ON/OFF/STOP

BESCHRIJVING
Dit is een opdracht voor sprite-evenementbehandeling. Het activeert (ON) of de-activeert (OFF/STOP) de detectie van botsingen tussen sprites in een BASIC-programma.
SPRITE ON moet worden uitgevoerd om de sprite-botsingdetectie te activeren, zodat de computer naar de subroutine springt die is aangegeven in de ON SPRITE GOSUB. Na SPRITE ON controleert de computer steeds op een sprite-botsing, voordat met een nieuwe opdracht wordt begonnen. Wordt de botsing gedetecteerd dan springt de computer onmiddellijk naar de subroutine. Na het uitvoeren van SPRITE STOP wordt gestopt met de detectie, maar er wordt wel onthouden dat er een botsing is geweest tijdens de SPRITE STOP; na het uitvoeren van SPRITE ON springt de computer opnieuw naar de sprite-subroutine.
SPRITE OFF beëindigt de sprite-detectie volledig. Ook wordt dan een botsing niet meer onthouden.

SYNTAX
SPRITE ON
SPRITE OFF
SPRITE STOP

VOORBEELDEN
In het volgende voorbeeld krijgt u twee vierkanten te zien, een gele en een witte, die elkaar van tegenovergestelde zijde op het beeldscherm naderen. Zodra ze botsen, wordt een ON SPRITE GOSUB uitgevoerd, zodat de subroutine wordt uitgevoerd. De SPRITE OFF in de sprite-onderbrekings-routine voorkomt iedere volgende detectie van sprite-botsingen.
(Probeer de routine ook eens uit zonder SPRITE OFF en let er op wat er gebeurt.)

```
10 ON SPRITE GOSUB 110
20 SCREEN 2,0
30 SPRITE$(0)=STRING$(8,CHR$(255))
40 SPRITE$(1)=STRING$(8,CHR$(255))
50 SPRITE ON
60 FOR I=10 TO 240
70 PUT SPRITE 0,(I,100),11,0
80 PUT SPRITE 1,(250-I,100),15,1
90 NEXT I
100 END
105 REM SPRITE-BOTSING-ROUTINE
110 SPRITE OFF
120 BEEP
130 RETURN
```

REGEL 10	geeft aan dat de detectie-subroutine vanaf regel 110 begint.
REGEL 20	zet het grafische scherm.
REGEL 30	sprite 0 is een vierkant.
REGEL 40	dit geldt ook voor sprite 1.
REGEL 50	sprite-botsingdetectie actief.
REGEL 60	de lus.
REGEL 70	sprite (geel) verplaatsen van links naar rechts.
REGEL 80	sprite (wit) verplaatsen van rechts naar links.
REGEL 90	volgende lus.
REGEL 100	einde.
REGEL 110	de-activeert de detectie (één keer is voldoende).
REGEL 120	signaal.
REGEL 130	keer terug in de lus.

PUNTEN OM TE ONTHOUDEN
De sprite-onderbreking wordt gede-activeerd als het programma niet meer draait en ook tijdens de foutdetectie-routines.

FOUTEN
U moet ON SPRITE GOSUB uitvoeren, alvorens u SPRITE ON/OFF/STOP wilt gebruiken. Anders zal de computer geen sprite-botsing detecteren.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ON SPRITE GOSUB
SPRITE$
PUT SPRITE
Sectie 2, Hoofdstuk 15: Geavanceerde grafische toepassingen IV: Grafische sprites

SPRITE$

BESCHRIJVING

U kunt een sprite definiëren met SPRITE$. U kunt tot 256 sprite-patronen hebben, indien u gebruik maakt van de sprite-grootte 0 of 1 (onvergroot), of 64 patronen bij sprite-grootte 2 of 3 (vergroot).
De lengte van een sprite is vast en is 32 bytes, maar voor kleinere sprites is het nodig alleen de eerste acht tekens te definiëren. De rest wordt automatisch gedefinieerd als CHR$(0).
Om een 16x16 sprite te definiëren moet u alle 32 bytes opgeven in het sprite-patroon.
De details over hoe sprites worden gebruikt vindt u in sectie 2, hoofdstuk 15: Sprites.

SYNTAX

SPRITE$(<integer>)=<string-uitdrukking>

VOORBEELDEN

```
10 SCREEN 2
20 SPRITE$(0)=CHR$(16)+CHR$(48)+CHR$(112)+
CHR$(255)+CHR$(255)+CHR$(112)+CHR$(48)+CHR$(16)
70 PUT SPRITE 0,(100,100),15,0
80 GOTO 80
```

REGEL 10	HIRES-schermmodus 2.
REGEL 20	definieert sprite 0.
REGEL 70	plaatst sprite 0 op positie X=100, Y=100, met witte kleur in sprite-vlak 0.

U ziet een pijl in het midden van het scherm.

PUNTEN OM TE ONTHOUDEN

U kunt sprites alleen gebruiken in schermmodus 1, 2 of 3. De sprite-grootte wordt bepaald met SCREEN. SCREEN overschrijft ervoor gedefinieerde sprites.
U kunt twee sprites met verschillende grootte niet mixen.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

PUT SPRITE
SCREEN
SPRITE ON/OFF/STOP
Sectie 2, Hoofdstuk 15: Geavanceerde grafische toepassingen IV: Grafische sprites

SQR

BESCHRIJVING
SQR is de vierkantswortelfunctie. Deze functie trekt de tweedemachtswortel uit een getal.
Het argument moet positief zijn. MSX-SQR kan geen complexe getallen aan.

SYNTAX
SQR(<numeriek>)

VOORBEELDEN

```
10 PRINT SQR(9)
20 PRINT SQR(ATN(1)*4)
```

```
RUN
3
1.7724538509055
```

PUNTEN OM TE ONTHOUDEN
SQR wordt berekend met dubbele nauwkeurigheid.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 19: Wiskundige functies

STEP

BESCHRIJVING
STEP is een onderdeel van de FOR...TO...NEXT-lus. STEP geeft aan met welke waarde de variabele in FOR...TO na elke iteratie moet worden verhoogd.
In STEP kunt u met kleinere waarden dan 1 werken, of zelfs met negatieve waarden. STEP 1 is overbodig omdat elke FOR...TO...NEXT zonder STEP standaard met 1 verhoogt.

SYNTAX
FOR <num var>=<numeriek> TO <numeriek> STEP <numeriek>

VOORBEELDEN

```
10 FOR I=1 TO 3 STEP 0.5
20 PRINT I;
30 NEXT I
40 PRINT
50 FOR J=3 TO 1 STEP -1
60 PRINT J;
70 NEXT J
```

```
RUN
1 1.5 2 2.5 3
3 2 1
```

FOUTEN
FOR I=1 TO 10 STEP -1
Dit is een veelgemaakte fout. Deze lus wordt slechts éénmaal doorlopen.

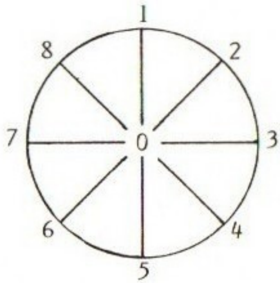
BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
FOR
TO
NEXT
Sectie 1, Hoofdstuk 5: Het gebruik van lussen

STICK joySTICK

BESCHRIJVING
STICK geeft de richting aan van de joystick of van de cursortoetsen, indien deze worden gebruikt als een joystick.
STICK(<n>) ... <n> kan zowel 0, 1 of 2 zijn.

n=0 cursortoetsen
n=1 joystick 1
n=2 joystick 2

Als de joystick in de neutrale stand staat, wordt 0 teruggegeven. Zoniet, dan geldt het volgende:



U dient twee cursortoetsen in te drukken om de diagonale richting aan te geven; richting 2 wordt bereikt door het tegelijkertijd indrukken van de cursortoetsen met de pijl naar boven en de pijl naar rechts.

SYNTAX
STICK(<n>)

VOORBEELDEN
Dit voorbeeld toont de richtingwaarden indien u de cursortoetsen als joystick gebruikt.

```
10 A=STICK(0)
20 LOCATE 20,10:PRINT A
30 GOTO 10
```

PUNTEN OM TE ONTHOUDEN
STICK geeft altijd een waarde tussen 0 en 8. Om de status te vinden van de

joystick-trekker moet u STRIG gebruiken.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
STRIG
Sectie 2, Hoofdstuk 23: Randapparatuur

STOP

BESCHRIJVING
STOP onderbreekt de BASIC-uitvoering en laat de computer terugkeren naar de commandomodus.
De 'Break in <regel>'-boodschap wordt afgedrukt na de STOP.

SYNTAX
STOP

VOORBEELDEN

```
10 PRINT 1909
20 STOP
```

```
RUN
1909
Break in 20
Ok
```

PUNTEN OM TE ONTHOUDEN
De STOP voert geen CLOSE uit voor eventueel geopende bestanden, hetgeen END wel doet.
De uitvoering kan met CONT worden voortgezet, vanaf de volgende regel.
U mag net zoveel STOP's gebruiken als u wenst.
STOP is duidelijk anders dan STOP ON/OFF/STOP.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
END
Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's

STOP ON/OFF/STOP

BESCHRIJVING
Deze opdracht is heel anders dan de STOP-opdracht, die BASIC onderbreekt. STOP ON/OFF/STOP activeert/de-activeert de detectie van <CTRL> <STOP>-toetsen. Als STOP ON wordt uitgevoerd, begint BASIC met de controle op het intoetsen van <CTRL> <STOP> bij iedere nieuwe opdracht. Blijkt inderdaad <CTRL> <STOP> te zijn ingetoetst dan wordt een subroutine aangeroepen die staat aangegeven in de ON STOP GOSUB-opdracht eerder in het programma.
Wordt STOP STOP uitgevoerd dan wordt niet direct naar de subroutine gesprongen indien <CTRL> <STOP> wordt ingetoetst, maar direct na een STOP ON wordt de subroutine aangeroepen.
STOP OFF zal de <CTRL> <STOP>-detectie volledig afstoppen.

SYNTAX
STOP ON
STOP OFF
STOP STOP

VOORBEELDEN
Het volgende programma laat zien hoe ON STOP GOSUB kan worden gebruikt om te voorkomen dat iemand in uw programma inbreekt. Regel 20 met STOP ON activeert de detectie. Drukt u vervolgens op <CTRL> <STOP> dan wordt een boodschap '<CTRL> <STOP> in-actief' afgegeven en het programma zet de uitvoering voort. Deze routine kent een speciale uitgangsroutine. Het drukken op de 's'-toets de-activeert de <CTRL> <STOP>-detectie. Anders wordt 'MSX' oneindig vaak afgedrukt.

```
10 ON STOP GOSUB 100
20 STOP ON
30 IF INKEY$="s" THEN STOP OFF:PRINT "<CTRL><STOP>
ACTIEF"
40 PRINT "MSX"
50 GOTO 30
100 BEEP
110 PRINT "<CTRL><STOP> IN-ACTIEF"
120 RETURN
```

REGEL 10	specificeert de positie van de <CTRL><STOP>-subroutine.
REGEL 20	activeert de detector.
REGEL 30	de-activeert de detectie en geeft een bericht af.
REGEL 40	drukt MSX af.
REGEL 50	sprong in de lus, terug naar 30
REGEL 100	geeft een waarschuwingssignaal.
REGEL 110	bericht.

REGEL 120 keert terug naar de plaats vanwaar de subroutine werd aangeroepen.

PUNTEN OM TE ONTHOUDEN

Let er op dat ON STOP-detectie de <STOP>-toets niet beïnvloedt. Alleen het inbreken in een programma wordt tegengehouden; dat is alles. Probeer <STOP> in te toetsen in het voorgaande programma. U zult zien dat het programma wordt gestopt en de cursor wordt getoond. Drukt u nogmaals op de <STOP>-toets dan gaat het programma verder.
Onthoud goed dat u eerst STOP ON moet uitvoeren om de <CTRL> <STOP>-detectie te activeren.
De <CTRL> <STOP>-interruptie wordt gede-activeert als het programma niet meer loopt en ook tijdens foutdetectieroutines.

FOUTEN

U moet een ON STOP GOSUB-opdracht hebben uitgevoerd en een <CTRL> <STOP>-routine hebben gedefinieerd.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

ON STOP GOSUB
Sectie 2, Hoofdstuk 9: Evenementbehandeling en interruptie door BASIC

STR$

BESCHRIJVING
STR$ geeft de string die behoort bij een numeriek argument.

SYNTAX
STR$(<numeriek>)

VOORBEELDEN

```
10 A$="PI="+STR$(3.14)
20 PRINT A$
```

```
RUN
PI=3.14
```

PUNTEN OM TE ONTHOUDEN
De tegengestelde functie van STR$ wordt uitgevoerd door VAL.

FOUTEN
Het argument moet numeriek zijn.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
VAL
Sectie 1, Hoofdstuk 15: Functies
Sectie 2, Hoofdstuk 3: Soort verandering

STRIG

BESCHRIJVING
Deze functie geeft de status af van de trekker of vuurknop van de joystick.

SYNTAX
STRIG(<n>)

<n>=0	spatiebalk gebruikt als trekker
<n>=1,3	joystick 1
<n>=2,4	joystick 2

STRIG(<n>)=0	niet ingedrukt
STRIG(<n>)=1	ingedrukt

VOORBEELDEN
Indien u de spatiebalk indrukt, verschijnt een bericht:

```
10 IF STRIG(0) THEN PRINT "SPATIEBALK INGEDRUKT..."
20 GOTO 10
```

PUNTEN OM TE ONTHOUDEN
STRIG wordt vooral in arcade-spellen toegepast.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
STICK
Sectie 2, Hoofdstuk 23: Randapparatuur

STRIG ON/OFF/STOP

BESCHRIJVING

Dit is iets anders dan de STRIG-opdracht, die de status afgeeft van één van de trekkertoetsen. STRIG(<n>) ON/OFF/STOP activeert/de-activeert de detectie van de trekkertoetsen. Na een STRIG ON zal, voordat een nieuwe BASIC-opdracht wordt uitgevoerd, een controle plaatsvinden op het al dan niet indrukken van één van de trekkertoetsen. Blijkt er een trekker te zijn ingedrukt dan wordt BASIC naar een met ON STRIG GOSUB aangegeven subroutine geleid.
Wordt een STRIG(<n>) STOP uitgevoerd dan springt de computer niet naar de subroutine, maar er wordt wel onthouden dat een trekker werd ingedrukt. Na een STRIG(<n>) ON zal de computer direct naar de subroutine springen.
STRIG(<n>) OFF beëindigt de detectie volledig.

SYNTAX

STRIG(<n>) ON
STRIG(<n>) OFF
STRIG(<n>) STOP
<n> is het trekkernummer.

Er zijn vijf trekkers:

0 = spatiebalk
1 = vuurknop voor joystick 1
2 = vuurknop voor joystick 2
3 = vuurknop voor joystick 1
4 = vuurknop voor joystick 2

VOORBEELDEN

Hier volgt een kort programma waarin de spatiebalk als trekker wordt gebruikt. Wordt de spatiebalk ingedrukt dan wordt de trekkerroutine aangeroepen. Zoniet, dan wordt voortdurend 'MSX' afgedrukt.

```
10 ON STRIG GOSUB 100
20 STRIG(0) ON
30 IF INKEY$="s" THEN END
40 PRINT "MSX"
50 GOTO 30
90 REM SPATIEBALK ALS TREKKERROUTINE
100 BEEP
110 PRINT "SPATIEBALK INGEDRUKT..."
120 RETURN
```

REGEL 10 geef aan waar de trekkersubroutine begint.

REGEL 20	activeer detectie.
REGEL 30	wordt <s> ingedrukt, stop dan.
REGEL 40	druk 'MSX' af.
REGEL 50	lus terug naar regel 30.
REGEL 100	waarschuwingssignaal.
REGEL 110	geeft een boodschap.
REGEL 120	keert terug naar de plaats van de aanroep.

```
RUN
MSX
MSX
MSX
MSX
SPATIEBALK INGEDRUKT...        Druk de spatiebalk in
MSX
MSX
MSX
Ok                             Druk op <s>
```

PUNTEN OM TE ONTHOUDEN
STRIG-interruptie wordt gede-activeert indien het programma niet meer loopt of tijdens de foutdetectieroutines.

FOUTEN
Een veel gemaakte fout is:

STRIG (0) ON...

Dit is fout. Er mag geen spatie staan tussen STRIG en (0).
U krijgt de boodschap 'Undefined Line Number' indien u een verkeerd regelnummer in ON STRIG GOSUB plaatst.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ON STRIG GOSUB
STRIG ON/OFF/STOP
Sectie 2, Hoofdstuk 9: Evenementbehandeling en interruptie door BASIC

STRING$

BESCHRIJVING
STRING$ geeft een string met een opgegeven aantal tekens.

SYNTAX
STRING$(<n>,<ASCII-waarde>)
Hiermee worden n tekens afgedrukt met de aangegeven <ASCII-waarde>.
STRING$(<lengte>,<string-var>)
Hiermee wordt een string afgedrukt bestaande uit <lengte> aantal tekens, die alle gelijk zijn aan het eerste teken in de stringvariabele.
<n> kan zowel een variabele als een constante zijn.

VOORBEELDEN

```
10 INPUT A
20 PRINT STRING$(A,"X")
30 GOTO 10
```

```
RUN
20
XXXXXXXXXXXXXXXXXXXX
10
XXXXXXXXXX
```

PUNTEN OM TE ONTHOUDEN
De STRING$-functie is handig bij het definiëren van vierkante sprites:

```
SPRITE$(0)=STRING$(8,CHR$(255))
```

FOUTEN
Wordt in STRING$ een string gebruikt waarvan de lengte groter is dan 200 tekens dan volgt een 'Out Of String Space'-foutmelding.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 20: De ASCII-codes

SWAP

BESCHRIJVING
Deze functie verwisselt de inhoud van twee variabelen.

SYNTAX
SWAP<variabele>,<variabele>

VOORBEELDEN

```
10 A=100:B=200
20 C$="MSX":D$="ASCII"
30 SWAP A,B
40 SWAP C$,D$
50 PRINT A,B
60 PRINT C$,D$
```

```
RUN
200        100
ASCII      MSX
```

FOUTEN
Probeert u verschillende soorten variabelen te verwisselen dan treedt een 'Type Mismatch'-foutmelding op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 6: Het gebruik van condities

TAB

BESCHRIJVING
TAB wordt altijd samen met PRINT of LPRINT gebruikt. TAB voert een tabulatie uit van een gegeven aantal posities, gerekend vanaf de linkerkant van het beeldscherm.

SYNTAX
PRINT TAB(<numeriek>)
LPRINT TAB(<numeriek>)

VOORBEELDEN

```
10 B=100
20 PRINT TAB(5)"B="TAB(10)B
```

```
RUN
    B=       100
Ok
```

PUNTEN OM TE ONTHOUDEN
<numeriek> moet kleiner zijn dan WIDTH-1 omdat anders naar de volgende regel wordt gesprongen.
Is <numeriek> een reëel getal dan wordt dit getal bij de decimale punt afgebroken om zo een integer te verkrijgen.
LOCATE is veel flexibeler dan TAB, omdat u X- en Y-coördinaten kunt meegeven.

FOUTEN
Is het argument groter dan 255 dan volgt een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
PRINT
LPRINT
LOCATE
Sectie 1, Hoofdstuk 9: Meer over PRINT en het beeldscherm

TAN

BESCHRIJVING
Geeft de tangens van het argument in radialen.. TAN wordt met dubbele nauwkeurigheid berekend.

SYNTAX
TAN(<numeriek>)

VOORBEELDEN
```
PRINT TAN(0.5)
 .54630248984381
```

FOUTEN
Is het argument niet numeriek (maar bijvoorbeeld een string) dan treedt een 'Type Mismatch Error' op.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
ATN
COS
SIN
Sectie 1, Hoofdstuk 19: Wiskundige functies

THEN

BESCHRIJVING
THEN wordt samen met IF gebruikt in de vaak voorkomende IF...THEN-opdracht. De opdrachten die volgen op THEN worden uitgevoerd indien de conditie na IF waar is.
Volgt er een regelnummer na THEN dan wordt dit herkend als een afgekorte vorm van GOTO <regelnummer>.
Er kunnen meerdere opdrachten, gescheiden door dubbele punten, volgen achter het woord THEN, indien u wenst dat de computer meerdere opdrachten uitvoert na een conditie die waar blijkt te zijn.

SYNTAX
IF <condities> THEN <opdrachten>
IF <condities> THEN <opdrachten> ELSE <opdrachten>
IF <condities> THEN <regelnummer>

VOORBEELDEN

```
10 INPUT X
20 IF X=10 THEN PRINT "JA"
30 GOTO 10
```

```
RUN
?10
JA
?
```

PUNTEN OM TE ONTHOUDEN
THEN kan worden vervangen door een GOTO-opdracht, maar alleen indien u naar een andere regel wenst te springen.

```
IF x=1 GOTO 2000
```

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
IF
GOTO
ELSE
Sectie 1, Hoofdstuk 6: Het gebruik van condities
Sectie 2, Hoofdstuk 7: BOOLEAN II: De IF...THEN...ELSE

TIME

BESCHRIJVING
De TIME-functie geeft de waarde van de interne systeemklok. De klok wordt op nul gezet zodra de machine wordt aangeschakeld en wordt automatisch verhoogd na een interruptie door de VIDEO DISPLAY PROCESSOR. Dit gebeurt 50 keer per seconde.
U kunt met de TIME-functie iedere integere waarde toekennen aan de klok. Ook is het mogelijk de klok op nul terug te zetten.

SYNTAX
TIME

VOORBEELDEN
Digitaal horloge:

```
10 CLS
20 TIME=0
30 MIN%=TIME/3600
40 SEC%=TIME/60-MIN%*60
50 LOVATE 10,11:PRINT "MINUTEN";MIN%
60 LOCATE 10,12:PRINT "SECONDEN";SEC%
```

Resultaat: U ziet een digitale klok in het midden van het scherm.

```
MINUTEN  1
SECONDEN 34
```

PUNTEN OM TE ONTHOUDEN
Als de VDP wordt uitgeschakeld stopt de klok. Dit kan gebeuren wanneer u gebruik maakt van een extern medium, zoals een cassetterecorder voor het laden van software.
TIME geeft altijd een positieve integer af.
TIME wordt toegepast in RND(-TIME) om werkelijk willekeurige getallen te genereren.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
RND
Sectie 1, Hoofdstuk 15: Functies

TO

BESCHRIJVING
TO is een onderdeel van de FOR...TO...STEP...NEXT-structuur en volgt altijd op FOR. De initiële waarde wordt gegeven vóór TO en de grenswaarde wordt gegeven ná TO.

SYNTAX
FOR <num var>=<numeriek> TO <numeriek>
FOR <num var>=<numeriek> TO <numeriek> STEP <numeriek>

VOORBEELDEN

```
10 FOR I=1 TO 2
20 PRINT I
30 NEXT I
```

```
RUN
1
2
OK
```

PUNTEN OM TE ONTHOUDEN
Een spatie is niet noodzakelijk tussen TO en de numerieke waarden, maar spaties maken het geheel wel beter leesbaar, bijvoorbeeld:

```
FOR I=100TO200
```

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
FOR
NEXT
STEP
Sectie 1, Hoofdstuk 5: Het gebruik van lussen

TROFF

BESCHRIJVING
Stopt TRON of stopt de trace (pluizer).
TROFF moet worden uitgevoerd om de trace (pluizer) te stoppen. Er is geen andere manier om TRON te beëindigen dan met TROFF.

SYNTAX
TROFF

PUNTEN OM TE ONTHOUDEN
Om een programma uit te pluizen kunt u TRON gebruiken.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
TRON
Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's

TRON

BESCHRIJVING
TRON zorgt ervoor dat de computer ieder regelnummer afdrukt van de op dat moment uit te voeren regel in een programma.
TRON is een pluizer en wordt gebruikt bij het opsporen van fouten in programma's. Dit zal vaak nodig blijken, met name wanneer uw programma een spaghetti-programma is geworden, zodat het op geen enkele andere manier te volgen is.

SYNTAX
TRON

VOORBEELDEN

```
10 FOR I=1 TO 2
20 PRINT I
30 NEXT I
```

```
TRON
OK
RUN
[10][20] 1
[30][20] 2
[30]
OK
```

PUNTEN OM TE ONTHOUDEN
TRON kan verwarring veroorzaken wanneer teveel regelnummers op het scherm verschijnen. TRON werkt niet in een grafische modus.
Om TRON uit te schakelen moet u TROFF gebruiken.

FOUTEN
Deze functie wordt gebruikt bij het opsporen van fouten.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
TROFF
Sectie 1, Hoofdstuk 7: Nuttige commando's en tips voor het schrijven van programma's.

USR

BESCHRIJVING
USR wordt gebruikt om vanuit een BASIC-programma een machinecodeprogramma aan te roepen. De positie van het machinecodeprogramma wordt bepaald met DEF USR.
U kunt met behulp van deze functie gegevens uitwisselen met de machinecode.

SYNTAX
USR[<cijfer>](<argument>)
<cijfer>=0 - 9
<argument>= iedere soort die moet worden overgedragen aan de machinecode.

VOORBEELDEN
```
PRINT USR0("QWERTY")
DMY=USR4(B%)
X=USR9(1000)
```

PUNTEN OM TE ONTHOUDEN
Het is niet mogelijk om hier aan te geven hoe u machinecode dient te gebruiken; raadpleeg de sectie over de USR-functie voor meer details.

FOUTEN
Indien u een eigen machinecoderoutine gebruikt, wees dan voorzichtig. Voer altijd eerst een CSAVE uit voordat u een programma uitvoert, omdat anders de machinecode het systeem kan verstoren en u geen mogelijkheid meer heeft om het programma terug te krijgen.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
DEF USR
Sectie 2, Hoofdstuk 21: USR-functie en machinecode

VAL

BESCHRIJVING
VAL geeft u de waarde van een string die een getal bevat. Is een string een voorstelling van een getal dan kan VAL worden uitgevoerd. Echter, VAL voor een uitdrukking werkt niet.
VAL negeert spaties, tabulatietekens en 'volgende regel'-tekens die vooraf kunnen gaan aan een getal (zie ook het voorbeeld). Ook met andere tekens die voorafgaan aan een getal kan VAL niet overweg, bijvoorbeeld VAL("ZXY100") levert geen 100 op.
VAL herkent plus- en mintekens.

SYNTAX
VAL(<string>)

VOORBEELDEN
```
PRINT VAL("   -100")
-100
PRINT VAL("RODEKOOL1000")
 0
A$="+10":PRINT VAL(A$)
 10
```

PUNTEN OM TE ONTHOUDEN
De tegenovergestelde functie van VAL wordt uitgevoerd door STR$.

FOUTEN
VAL("-100+900") geeft alleen -100 terug, omdat VAL geen uitdrukkingen in een string aankan.
VAL(A%) veroorzaakt een 'Type Mismatch'-fout.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
STR$
Sectie 1, Hoofdstuk 15: Functies
Sectie 2, Hoofdstuk 3: Soortverandering

VARPTR

BESCHRIJVING
VARPTR(<naam van variabele>) geeft u het adres terug van de eerste byte van de gegevens, aangegeven door de naam van de variabele. De naam van de variabele kan van iedere soort zijn, maar het zal duidelijk zijn dat de variabele wel moet bestaan alvorens u VARPTR gebruikt.
Ook kan het adres van het begin van het bestandsbesturingsblok worden verkregen.
Het teruggegeven adres zal een integer zijn, in de reeks -32768 t/m 32767. Wordt een negatief getal gegeven, voeg hieraan dan 65536 toe om het eigenlijke adres te verkrijgen.

SYNTAX
VARPTR(<naam van variabele>)
VARPTR(#<bestandsnaam>)

VOORBEELDEN
```
PRINT VARPTR(A(0))
S$="||||":PRINT 65536+VARPTR(S$)
D%=100:PRINT "H";HEX$(65536+VARPTR(D%))
```

PUNTEN OM TE ONTHOUDEN
VARPTR wordt soms gebruikt om het beginadres te verkrijgen van een array, opdat dit kan worden overgedragen aan een machinecoderoutine. Een functie-aanroep in de vorm VARPTR(A(0)) wordt gewoonlijk gebruikt bij het overdragen van een array, opdat het laagste adres van de array wordt verkregen.

FOUTEN
Indien de variabele waarnaar wordt verwezen niet bestaat dan volgt een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 2, Hoofdstuk 20: Geheugenorganisatie

VDP Video Display Processor

BESCHRIJVING

VDP wordt gebruikt om toegang te verkrijgen tot de Video Display Processor.
Er zijn negen registers.
0-7 zijn write-only; dit betekent dat u er een waarde aan kunt toekennen.
8 is een read-only; deze kunt u dus alleen lezen.
De details over de VDP-registers vindt u uitvoerig in de grafische sectie: Hoe toegang wordt verkregen tot de Video Display Processor.

SYNTAX

Voor registers 0-7, <cijfer>=0 - 7
VDP(<cijfer>)=<numeriek>

Register 8
<num var>=VDP(8)

VOORBEELDEN

```
PRINT "VDP STATUS REGISTER";BIN$(VDP(8))
```

Bovenstaande regel drukt de binaire weergave af van statusregister 8.

PUNTEN OM TE ONTHOUDEN

De VDP-processor van de NL-versie van MSX is de TMS 9929A die past bij het PAL-TV-systeem.

FOUTEN

Misbruikt u de VDP dan kunt u het beeld op het beeldscherm verstoren. Schakel dan uit en start opnieuw.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN

Sectie 2, Hoofdstuk 16: Geavanceerde grafische toepassingen. V: Hoe toegang wordt verkregen tot de VIDEO DISPLAY PROCESSOR

VPEEK Video-RAM PEEK

BESCHRIJVING
VPEEK leest in de Video-RAM.

SYNTAX
VPEEK(<adres>)
<adres>=0 t/m 16383

VOORBEELDEN
```
PRINT VPEEK(100)
V%=VPEEK(999)
```

PUNTEN OM TE ONTHOUDEN
Het is aan te bevelen dat alleen ervaren gebruikers deze functie aanwenden.
Omdat de Video-RAM is gescheiden van het hoofdgeheugen, dient u deze functie te gebruiken om de RAM ervan te lezen.
Alle MSX-computers beschikken over 16 Kbytes Video-RAM.
VPEEK is de complementfunctie van VPOKE.

FOUTEN
Valt <adres> buiten de toegestane waarden dan volgt een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
VPOKE
Sectie 2, Hoofdstuk 17: Geavanceerde grafische toepassingen VI: De Video-RAM

VPOKE Video-RAM POKE

BESCHRIJVING
VPOKE schrijft een waarde in de Video-RAM. U dient hierbij het adres van de Video-RAM mee te geven en de weg te schrijven gegevensbyte.
De waarde van de byte moet liggen tussen 0 en 255.

SYNTAX
VPOKE <adres>,<byte>
<adres>= reeks 0 t/m 16383
<byte>= reeks 0 t/m 255

VOORBEELDEN
```
VPOKE 998,255
```

PUNTEN OM TE ONTHOUDEN
Gebruik deze functie alleen als u weet wat u doet.
Omdat de Video-RAM is gescheiden van het hoofdgeheugen heeft u deze speciale POKE-functie nodig om toegang te verkrijgen.
Alle MSX-computers beschikken over 16 Kbytes Video-RAM.
VPOKE is de complementfunctie van VPEEK.

FOUTEN
Indien <adres> buiten de toegestane waarden valt treedt een 'Overflow Error' op.
Ligt de waarde van <byte> buiten de toegestane waarden dan volgt een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
VPEEK
Sectie 2, Hoofdstuk 7: Geavanceerde grafische toepassingen VI: De Video-RAM

WAIT

BESCHRIJVING
WAIT vertraagt de programma-uitvoering en wacht totdat een gespecificeerde poort een bepaald bit-patroon ontwikkelt.

SYNTAX
WAIT<poortnummer>,<I>,[<J>]

Reeks:
<poortnummer>=0 t/m 255
<I> en <J>=0 t/m 255

PUNTEN OM TE ONTHOUDEN
De ingelezen gegevens bij de poort worden geXORed met J en geAND met I. Is het resultaat 0 dan keert BASIC terug en leest opnieuw gegevens in bij de poort. (Wordt J weggelaten dan is J=0.)
Is het resultaat ongelijk aan 0 dan begint de programma-uitvoering opnieuw.
Deze opdracht heeft toegang tot de I/O-poorten, zonder daarbij de BIOS te raadplegen. U loopt daarom het risico dat, indien u deze opdracht gebruikt, u afwijkt van de MSX-standaard en machine-afhankelijk wordt.
MSX gebruikt alleen de poorten 0 t/m 255: alles boven 255 heeft geen betekenis.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
OUT
INP

WIDTH

BESCHRIJVING
WIDTH bepaalt de breedte van het scherm. Bij de uitvoering ervan wordt het scherm gewist.

SCREEN MODUS	VERVANGINGSWAARDE	MAXIMUM
schermmodus 0	37	40
schermmodus 1	29	32

SYNTAX
WIDTH <numeriek>

VOORBEELDEN
```
WIDTH 30
WIDTH 20
```

PUNTEN OM TE ONTHOUDEN
De minimumbreedte is 1.

FOUTEN
Valt de waarde van het argument buiten de toegestane waarden dan volgt een 'Illegal Function Call'.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
Sectie 1, Hoofdstuk 21: Schermmodus

XOR

BESCHRIJVING
XOR (EXCLUSIVE OR) is één van de logische bewerkers die BOOLEAN algebra uitvoert.

XOR	X	X	X XOR Y
	0	0	0
	1	0	1
	0	1	1
	1	1	0

Daarom wordt 100 XOR 50 als volgt berekend:

	X	100	0000000001100100
XOR	Y	50	0000000000110010
		86	0000000001010110

SYNTAX
<num var>=<numeriek> XOR <numeriek>

PUNTEN OM TE ONTHOUDEN
De volgende relatie is waar:

A = A XOR Y XOR Y

FOUTEN
Er treedt een 'Overflow Error' op indien het argument buiten de toegestane waarden valt.

BIJBEHORENDE SLEUTELWOORDEN EN VERWIJZINGEN
AND
OR
EQV
NOT
IMP
Sectie 2, Hoofdstuk 6: BOOLEAN algebra

Deel 4
Het besturingssysteem

Sectie 4

Het besturingssysteem

Het volgende gedeelte behandelt de BIOS (Basic Input/Output System, of Nederlands: Basis Invoer/Uitvoer Systeem). Er wordt uitgelegd hoe de diverse systeemroutines kunnen worden gebruikt en er wordt een overzicht gegeven van de systeem-RAM.
Om dit gedeelte van het boek goed te begrijpen, dient de lezer kennis te hebben van de Z80 assembleertaal. Er wordt hier geen poging gedaan u dat te leren.
De secties 4.1 tot 4.7 beschrijven de diverse BIOS-aanroepen die beschikbaar zijn op iedere MSX-computer. Aan elkaar verwante aanroepen worden samen gegroepeerd in ieder van de secties. Het formaat van de BIOS-aanroepbeschrijving is gestandaardiseerd. De eerste regel bevat de naam van de routine, gevolgd door het aanroepadres in hexadecimale vorm. De volgende regel geeft de Z80-instructie die moet worden uitgevoerd om de BIOS-routine aan te roepen. Hierna volgt een verklaring van de functie van de aanroep en gegevens over hoe parameters kunnen worden uitgewisseld tussen de BIOS-routines.
Het daaropvolgende gedeelte geeft een lijst van de Z80-registers en geheugenlocaties die aan verandering onderhevig zijn in de betreffende routine. Ten slotte volgt een gedeelte over de status van de interrupties en mogelijke toepassingen van de routine.
Sectie 4.8 verklaart het gebruik van zgn. hooks, gevolgd door een alfabetisch overzicht van iedere hook met beginadres en vanwaar deze wordt aangeroepen.
Het laatste gedeelte van dit hoofdstuk bevat een alfabetische lijst van systeem-RAM-locaties, met de betreffende adressen en lengtes (in bytes).

Sectie 4.1

RST-instructies

Dit gedeelte behandelt de acht instructies die kunnen worden uitgevoerd met de Z80-RST-instructies:

RST0	CHKRAM
RST1	SYNCHR
RST2	CHRGTR
RST3	OUTDO
RST4	DCOMPR
RST5	GETYPR
RST6	CALLF
RST7	KEYINT

CHKRAM 0000

Uitgevoerd via RST 00.
Na het aanschakelen van de computer begint hier de eerste uitvoering. Het springen naar dit adres veroorzaakt een volledige initialisatie van het systeem (reset). De naam CHKRAM verwijst naar een controle van de RAM, hetgeen misleidend is omdat deze routine aanmerkelijk meer doet dan controleren.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Deze zijn niet van toepassing omdat de computer nooit terugkeert na een aanroep van dit adres.

Gewijzigde registers en geheugenvelden
Alle registers worden gewijzigd en alles in het geheugen gaat verloren.

Mogelijke toepassingen
Er bestaat een toepassing van deze routine - het initialiseren van de computer (of kortweg 'resetten').

SYNCHR 0008

Uitgevoerd via RST1.

Deze routine wordt gebruikt door de BASIC-vertolker om op syntax-fouten te controleren. Wordt geen fout ontdekt dan wordt de uitvoering voortgezet met een aanroep van CHRGTR (0010). Blijkt er iets fout te zijn dan volgt een 'Syntax Error'.

Invoerparameters
HL is de verwijzing naar het volgende teken van de BASIC-tekst en de byte die volgt op RST1 bevat de waarde waarmee wordt vergeleken.

Uitvoerparameters
Na terugkeer verwijst HL naar het volgende teken en A bevat dit teken. de 'carry flag' wordt gezet indien het teken een getal is en de 'zero flag' wordt gezet indien het einde van de opdracht is bereikt.

Gewijzigde registers en geheugenvelden
De accumulator, de flags en het HL-registerpaar worden aangetast door deze aanroep.

Opmerkingen
Deze aanroep heeft geen betekenis voor machinecodeprogrammeurs en kan beter niet worden gebruikt.

CHRGTR 0010

Uitgevoerd via RST2.
Deze wordt aangeroepen via de BASIC-vertolker om het volgende teken uit het BASIC-programma op te halen. Gewoonlijk verloopt de aanroep via de routine SYNCHR (RST1).

Invoerparameters
HL moet verwijzen naar het op te halen teken.

Uitvoerparameters
Na terugkeer verwijst HL naar het volgende teken in het programma. De 'carry flag' wordt gezet indien een getal werd gelezen en de 'zero flag' wordt gezet indien het einde van een opdracht is bereikt.

Gewijzigde registers en geheugenvelden
AF en HL worden aangetast door deze RST.

Opmerkingen
Evenals SYNCHR (RST1) kan deze routine beter niet worden gebruikt.

OUTDO 0018

Uitgevoerd via RST3.
Deze instructie schrijft de inhoud van de accumulator naar het op dat moment geselecteerde medium (of randapparaat).

Invoerparameters
De accumulator moet de code bevatten die wordt weggeschreven. Wordt de code geschreven naar een disk-bestand dan dient PTRFIL te verwijzen naar dat bestand. Moeten de gegevens naar een printer worden gestuurd dan dient PTRFIL een nul te bevatten en PTRFLG moet ongelijk aan nul zijn. Moet de code naar de terminal worden verstuurd dan dienen PTRFIL en PRTFLG beide nul te zijn.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
Er worden geen registers of geheugenvelden gewijzigd door deze aanroep.

Opmerkingen
Na het plaatsen van AF op de stack roept deze routine de hook H.OUTD aan, voordat met de uitvoering wordt verdergegaan.

DCOMPR 0020
Uitgevoerd via RST4.
Deze aanroep vergelijkt de HL- en DE-registerparen van de Z80.

Invoerparameters
Er zijn geen invoerparameters benodigd, behalve de HL- en DE-registerparen.

Uitvoerparameters
Na terugkeer wordt de 'carry flag' gezet indien HL minder is dan DE, anders wordt deze gewist. Is HL=DE dan wordt de 'zero flag' gezet.

Gewijzigde registers en geheugenvelden
Alleen de accumulator en de flags worden gewijzigd door deze instructie.

GETYPR 0028
Uitgevoerd via RST5.
Deze instructie wordt door de BASIC-vertolker gebruikt om te bepalen welke soort drijvende-komma-accumulator er wordt gebruikt.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
De verschillende flags worden gezet, afhankelijk van de soort drijvende-komma-accumulator die op dat moment wordt gebruikt.

Gewijzigde registers en geheugenvelden
Alleen de flags worden aangetast door deze routine.

CALLF 0030

Uitgevoerd via RST6.
RST voert een zgn. interslotaanroep uit.

Invoerparameters
De byte die volgt op de RST6-instructie bevat de beschrijving van het slot en de daaropvolgende twee bytes bevatten het aan te roepen adres.

Invoerparameters
Deze hangen af van de aan te roepen routine.

Gewijzigde registers en geheugenvelden
Deze hangen af van de aan te roepen routine.

Opmerkingen
De slotbeschrijving heeft de volgende vorm:

bitnr.	7	6	5	4	3	2	1	0
inhoud	F	x	x	x	S	S	P	P

waarbij:

PP	(0-3)	is het nummer van het primair te selecteren slot.
SS	(0-3)	is het nummer van het secundaire slot.
F	(0-1)	is een flag die op 1 wordt gezet indien de secundaire slots in gebruik zijn. Ze wordt op 0 gezet als dat niet het geval is.

Bits 4-6 worden niet gebruikt en kunnen zowel op- als afgezet worden.

KEYINT 0038

Uitgevoerd via RST7.
Het MSX-systeem opereert in de interruptiemodus 1, dus dit is de RST-instructie die bij elke gemaskeerde interruptie optreedt. Interrupties treden iedere 0.02 seconde op vanwege de 50 Hz-timer. Indien dit gebeurt wordt de intervalteller verlaagd, FIFFY verhoogd, de muziekwachtrijen worden aangepast, de joystick-trekkers worden gecontroleerd en het toetsenbord wordt afgetast.

Invoerparameters
Door zijn natuur (dit is de hardware-interruptie) kan er geen sprake zijn van invoerparameters.

Uitvoerparameters
Verschillende detecties kunnen worden aangegeven in de flags (bijv. ON SPRITE) en een andere toets kan worden geplaatst in het toetsenbordgeheugen.

Gewijzigde registers en geheugenvelden
Alle registers blijven ongewijzigd, maar veel systeemgeheugen wordt aangetast door deze aanroep.

Opmerkingen
De hook H.KEYI wordt direct aangeroepen na het plaatsen van de registers op de stack, zodat andere interrupties kunnen worden verwerkt.

Routines die betrekking hebben op het behandelen van slots

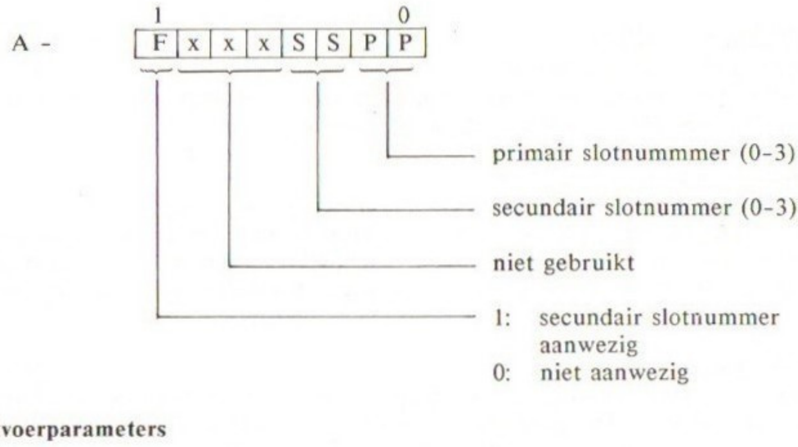
De BIOS-routines die in dit gedeelte worden beschreven, worden gebruikt om het slotssysteem (zie voor details sectie 2, hoofdstuk 22: MSX-geheugenorganisatie en cartridge-slotmechanisme) te besturen.

RDSLT 000C

Deze routine wordt gebruikt om een geheugenlocatie in te lezen in een willekeurig slot.

Invoerparameters

Het HL-registerpaar moet het adres bevatten van de in te lezen locatie en de accumulator moet het slot aangeven waarvan wordt gelezen. De waarde in de accumulator heeft de volgende betekenis: de laagste twee bits bevatten het primaire slotnummer (0-3), de volgende twee bits (0-3) het secundaire slotnummer en de laatste drie bits worden niet gebruikt; is een secundair slotnummer meegegeven dan dient de topbit te zijn gezet. Als dat niet het geval is dient deze te worden gewist.



Uitvoerparameters

De accumulator bevat de inhoud van de aangegeven geheugenlocatie.

Gewijzigde registers en geheugenvelden
De registers AF, BC en DE worden aangetast door deze routine.

Opmerkingen
Deze routine de-activeert alle interrupties. Een poging om pagina 3 van een slot te selecteren is er de oorzaak van dat de computer gaat hangen! (Zie ENASLT om dit probleem te verhelpen.)

WRSLT 0014

Uitgevoerd via CALL &H0014.
Deze aanroep wordt gebruikt om in een willekeurige geheugenlocatie te schrijven in een willekeurig slot.

Invoerparameters
Het HL-registerpaar bevat het adres van de locatie waarnaar moet worden geschreven; de accumulator geeft het te lezen slotnummer aan en register E het getal dat moet worden geschreven. De inhoud van de accumulator heeft hetzelfde formaat zoals dat is beschreven in RDSLT (000C).

Uitvoerparameters
Er worden geen parameters teruggegeven door deze routine.

Gewijzigde registers en geheugenvelden
De registers AF, BC en DE worden door de aanroep aangetast.

Opmerkingen
Deze routine de-activeert alle interrupties. Een poging om met deze routine pagina 3 van een slot te selecteren heeft tot gevolg dat het systeem gaat hangen! (Zie ENASLT om dit probleem te verhelpen.)

CALSLT 001C

Uitgevoerd via CALL &H001C
Deze routine voert een inter-slotaanroep uit, selecteert een pagina in een ander slot en roept hierin vervolgens een adres aan.

Invoerparameters
Het IX-registerpaar moet het adres bevatten dat wordt aangeroepen en in de meest significante bit van het IY-registerpaar wordt het slotnummer aangegeven. Het formaat hiervan is identiek aan dat van de accumulator, zoals beschreven in RDSLT. Parameters kunnen worden overgedragen via AF, BC, DE en HL, maar niet via de alternatieve registers.

Uitvoerparameters
Er worden geen parameters teruggegeven, tenzij zij zijn gecreëerd door een inter-slotaanroep. De parameters kunnen in willekeurige registers worden teruggegeven, behalve in het alternatieve accumulatorregisters, omdat deze de status bevat van het slotselectie-mechanisme voor de aanroep.

Gewijzigde registers en geheugenvelden
De registers AF', BC', DE' en HL' worden gewijzigd voordat de interslotaanroep is uitgevoerd.

Opmerkingen
Interrupties worden gede-activeerd door deze routine. Deze aanroep kan ook worden verkregen via een RST6-instructie (zie CALFF, sectie 2.1). Een poging om pagina 3 van een slot te selecteren heeft tot gevolg dat het systeem gaat hangen! (Zie ENASLT om dit probleem te verhelpen.)

ENASLT 0024

Uitgevoerd via CALL &H0024.
Deze aanroep selecteert een pagina van één van de slots.

Invoerparameters
De meest significante twee bits van het H-register worden gebruikt om de juiste pagina te selecteren, en wel als volgt:

MSB	geselecteerde pagina
00	0000-3FFF
01	4000-7FFF
10	0000-BFFF
11	C000-FFFF

De accumulator geeft het te selecteren slotnummer aan, zoals aangegeven bij RDSLT.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL worden aangetast door deze aanroep.

Opmerkingen
Deze routine de-activeert alle interrupties.
Het probleem waarbij pagina 3 van een slot niet kan worden geselecteerd bij de aanroepen RDSLT, WRSLT en CALSLT, kan worden verholpen met deze routine. Bijvoorbeeld om adres &HD000 in slot 3 te lezen kunt u de volgende codering gebruiken:

```
CALL &H0138     : lees het primaire selectieslotregister
PUSH AF         : bewaar het bovenstaande
LD HL,&HD000    : het te lezen geheugenadres
PUS HL          : bewaar dit adres
LD A,3
DI              : de interrupties moeten onmogelijk worden gemaakt,
                  omdat zij pagina 3 beïnvloeden
CALL  &H0024    : activeer slot 3, pagina 3
```

```
POP H,(HL)      : neem gewenste inhoud
CALL &H013B     : activeer systeem pagina 3
EI
LD A,H          : geef de gewenste waarde in de accumulator
RET
```

Identieke methodes kunnen worden gebruikt om in pagina 3 van een ander slot te schrijven op een routine aan te roepen.

RSLREG 0138

Uitgevoerd via CALL &H0138.
Deze aanroep leest de inhoud van het primaire slotselectieregister.

Invoerparameters
Er zijn geen invoerparameters nodig.

Uitvoerparameters
Een kopie van het primaire slotselectieregister wordt geplaatst in de accumulator.

Gewijzigde registers en geheugenvelden
Alleen de accumulator wordt door deze aanroep gewijzigd.

Opmerkingen
Deze routine is eenvoudigweg een IN-instructie (IN&HAB) wellicht), gevolgd door een RET.

WSLREG 013B

Uitgevoerd via CALL &H013B.
Deze routine schrijft naar het primaire slotselectieregister.

Invoerparameters
De accumulator dient de te schrijven waarde te bevatten.

Uitvoerparameters
Er worden geen parameters teruggegeven door deze routine.

Gewijzigde registers en geheugenvelden
Deze worden niet gewijzigd.

Opmerkingen
Deze routine is eenvoudigweg een OUT-instructie (OUT &HA8 wellicht), gevolgd door een RET.

CALBAS 0159

Uitgevoerd via CALL &H0159.
Deze aanroep wordt gebruikt door de BASIC-vertolker om een inter-slotaanroep naar de BASIC-vertolker-uitbreidingscartridge uit te voeren.

Invoerparameters
Het aan te roepen adres dient in het IX-registerpaar te staan.

Uitvoerparameters
De uitvoer van deze aanroep hangt af van de inter-slotaanroep.

Gewijzigde registers en geheugenvelden
Evenals hierboven is dit niet gedefinieerd.

Sectie 4.3

BIOS-routines om toegang te verkrijgen tot de console

Dit gedeelte beschrijft de routines die worden gebruikt om de 'console' te besturen, dat wil zeggen het toetsenbord, het tekstbeeldscherm en de printer.

CHSNS 009C

Uitgevoerd via CALL &H009C.
Deze aanroep voert twee bewerkingen uit. Ten eerste wordt de status van de <SHIFT>-toetsen gelezen. Is een <SHIFT>-toets ingedrukt en het functietoetsbeeld actief, dan worden de functietoetsen 6-10 getoond. Ten tweede controleert deze routine de status van de toetsenbordbuffer.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Blijkt de toetsenbordbuffer leeg te zijn dan wordt de zero-flag gezet, anders wordt deze gewist.

Gewijzigde registers en geheugenvelden
Alleen de accumulator en de flags worden aangetast door deze routine.

Opmerkingen
Interrupties worden ge-activeerd door deze routine.

CHGET 009F

Uitgevoerd via CALL &H009F.
Deze routine geeft het teken van de toetsenbordbuffer. Is de buffer leeg, dan wordt gewacht op een toetsindruk. Is de cursor actief dan wordt deze tijdens het wachten getoond.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
In de accumulator wordt de code geplaatst van het teken dat behoort bij de ingedrukte toets.

Gewijzigde registers en geheugenvelden
Alleen de accumulator en de flags worden aangetast door deze routine.

Opmerkingen
Deze routine roept ook H.CHGE aan, direct nadat de respectievelijke registers HL, DE en BC op de stack zijn geplaatst. Dit maakt het mogelijk dat andere 'console'-apparaten kunnen worden gebruikt.

CHPUT 00A2

Uitgevoerd via CALL &H00A2.
Deze aanroep stuurt een teken naar de 'console' en verplaatst daarbij naar de volgende regel en scrollt het scherm indien nodig. De besturingstekens 7-13 en 27-31 kunnen ook worden verstuurd.

Invoerparameters
De accumulator bevat de code van het te versturen teken. De positie van de cursor op het scherm wordt opgeslagen in de geheugenlocaties CSRX, CSRY.

Uitvoerparameters
Na het beëindigen van de routine zal de cursorpositie (CSRX, CSRY) worden aangepast.

Gewijzigde registers en geheugenvelden
Er worden geen registers gewijzigd, maar de geheugenposities CSRX en CSRY kunnen zijn veranderd.

Opmerkingen
Na het plaatsen van de registers HL, DE, BC en AF op de stack, roept deze routine hook H.CHPU aan, zodat ook andere 'console'-apparaten kunnen worden gebruikt (bijv. 80-kolommenscherm). Deze routine voert niets uit, indien de grafische modus II of de meerkleurenmodus actief is.
Interrupties worden ge-activeerd door deze aanroep.

LPTOUT 00A5

Uitgevoerd via CALL &H00A5.
Deze routine verstuurt een teken naar de printer, op voorwaarde dat deze is aangesloten. Is de printer niet gereed ('ready') om een teken te ontvangen dan blijft de routine wachten totdat deze wél gereed is, of tot de <STOP>-toets wordt ingedrukt.

Invoerparameters
Het af te drukken teken moet in de accumulator worden geplaatst.

Uitvoerparameters
Na terugkeer wordt de 'carry flag' gewist indien de afdruk van het teken met succes is gebeurd en gezet indien de routine is onderbroken door de <STOP>-toets.

Gewijzigde registers en geheugenvelden
De flags kunnen zijn veranderd door deze aanroep.

Opmerkingen
Allereerst roept deze routine hook H.LPTO aan, zodat aanvullende processen kunnen worden uitgevoerd. Een voorbeeld hiervan is: de hook zo te programmeren dat tekens die worden verstuurd door de printer worden genegeerd, via:

```
H.LPTO:  INC   SP
         INC   SP
         RET
```

Hook H.LPTO bevindt zich op adres &HFFB6, zodat bovenstaand voorbeeld kan worden geprogrammeerd met behulp van de BASIC-opdrachten:

```
POKE &HFFB6,&H33
POKE &HFFB7,&H33
```

(Om de printer weer in werking te krijgen dient u opnieuw in te tikken: POKE &HFFB6,&HC9.)

LPTSTT 00A8

Uitgevoerd via CALL &H00A8.
Deze aanroep controleert of de printer gereed is om een teken te ontvangen.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Is de printer gereed om gegevens te ontvangen dan bevat de accumulator 255 en de 'zero-flag' zal worden gewist, anders bevat de accumulator nullen en is de 'zero-flag' gezet.

Gewijzigde registers en geheugenvelden
Alleen de accumulator en de flags worden aangetast door deze routine.

Opmerkingen
LPTSTT wordt aangeroepen door LPTOUT.
Deze routine begint met een aanroep van hook H.LPTS.

CNVCHR 00AB

Uitgevoerd via CALL &H00AB.

Deze aanroep zet een tekencode om naar een getal van het patroon die het vertegenwoordigt op het scherm van de Video Display Processor. Voor de tekencodes 32-255 zijn deze getallen gelijk aan de patroonnummers voor dat teken, echter de tekencodes 0-31 zijn besturingstekens en de patroonnummers 0-31 stellen de tekens voor die worden verkregen door het indrukken van de grafische toets. Deze patronen worden vertegenwoordigd door twee tekencodes - besturingsteken 1, gevolgd door een teken in de reeks 64-95.

Invoerparameters

Het aan te passen teken dient in de accumulator te worden gezet.

Uitvoerparameters

Er kunnen vier verschillende resultaten worden verkregen met deze aanroep, afhankelijk van de initiële status van de grafische hoofdbyte (GRPHED) en de accumulator.

1. Is GRPHED gelijk aan nullen en bevat de accumulator tekencode 1, dan blijft de accumulator ongewijzigd, GRPHED wordt op 1 gezet en de 'carry'- en 'zero-flag' worden gewist.
2. Is GRPHED gelijk aan nullen en bevat de accumulator een waarde ongelijk aan tekencode 1, dan blijven GRPHED en de accumulator ongewijzigd en de C en Z worden gezet (resp. 'carry'- en 'zero flag').
3. Is GRPHED ongelijk aan nullen en bevat de accumulator een waarde van 64-95, dan zal na terugkeer de accumulator 64 lager zijn dan de initiële waarde en GRPHED zal nullen bevatten, C wordt gezet en Z wordt gewist.
4. Is GRPHED ongelijk aan nullen en bevat de accumulator een waarde van 64-95, dan wordt GRPHED op nullen gezet, de accumulator blijft ongewijzigd en C en Z worden gezet.

Gewijzigde registers en geheugenvelden

De accumulator, de flags en de geheugenlocatie-GRPHED kunnen worden gewijzigd door deze aanroep.

PINLIN 00AE

Uitgevoerd via CALL &H00AE.

Deze routine is gelijk aan INLIN, maar indien de automatische regelnummering actief is, zal het indrukken van <CTRL-U> niet ook het regelnummer verwijderen.

Invoerparameters

Deze zijn niet nodig.

Uitvoerparameters

Het HL-registerpaar verwijst naar de geheugenlocatie onder het eerste teken in de buffer en de 'carry-flag' wordt gezet indien <CTRL> <STOP> werd ingedrukt.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL kunnen worden veranderd door deze aanroep.

INLIN 00B1

Uitgevoerd via CALL &HOOB1.
Deze routine wordt door de BASIC-vertolker gebruikt om een regel vanaf het toetsenbord te accepteren, de tekens worden op het beeldscherm afgebeeld zoals ze zijn ingetoetst en de regel wordt opgeslagen in een buffer, totdat <RETURN> of <CTRL> <STOP> wordt ingedrukt.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Het HL-registerpaar verwijst naar de geheugenlocatie onder het eerste teken in de buffer en de 'carry-flag' wordt gezet indien <CTRL> <STOP> wordt ingedrukt.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL kunnen worden gewijzigd door deze routine.

QUINLIN 00B4

Uitgevoerd via CALL &HOOB4.
Deze routine is gelijk aan INLIN, maar wordt gebruikt door de INPUT-opdracht om een vraagteken af te drukken en vervolgens een regel te accepteren vanaf het toetsenbord.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Het HL-registerpaar verwijst naar de geheugenlocatie onder het eerste teken in de buffer en de 'carry-flag' wordt gezet indien <CTRL> <STOP> wordt ingedrukt.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL kunnen worden gewijzigd door deze routine.

BREAKX 00B7

Uitgevoerd via CALL &HOOB7.
Deze aanroep controleert of <CTRL> <STOP> wordt ingedrukt.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Indien <CTRL> <STOP> wordt ingedrukt, wordt de 'carry-flag' gezet, anders wordt deze gewist.

Gewijzigde registers en geheugenvelden
De accumulator en de flags worden aangetast door deze routine.

Opmerkingen
Deze routine zal alleen worden gebruikt indien de interrupties zijn gedeactiveerd, omdat gewoonlijk <CTRL> <STOP> wordt onderkend door de 50 Hz-interruptie.

ISCNTC 00BA

Uitgevoerd via CALL &H00BA.
Deze aanroep controleert of <STOP> of <CTRL> <STOP> wordt ingedrukt. Wordt <STOP> ingedrukt dan wordt een programmalus uitgevoerd totdat opnieuw <STOP> wordt ingedrukt. Wordt <CTRL> <STOP> ingedrukt en werd dit nog niet eerder onderkend, dan voert de computer de onderbrekingsroutine uit, dit is terugschakelen naar de tekstmodus, het activeren van het slot met de BASIC-vertolker en ten slotte springen naar de BASIC-commandomodus. Wordt noch <STOP> noch <CTRL> <STOP> ingedrukt dan voert de routine niets uit.

Invoerparameters
Invoerparameters zijn niet benodigd voor deze routine, de 50 Hz-timer kan er echter voor hebben gezorgd dat INTFLG een 3 bevat indien reeds <CTRL> <STOP> werd ingedrukt en 4 indien <STOP> werd ingedrukt.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
De accumulator, de flags en de geheugenlocatie INTFLG en KILBUF kunnen worden gewijzigd door deze aanroep.

Opmerkingen
Allereerst inspecteert deze routine de locatie BASROM. Is deze nul dan vervolgt de routine zoals hierboven beschreven en anders wordt teruggekeerd.
Daarom heeft het plaatsen van een waarde ongelijk aan nul in BASROM tot gevolg dat de BASIC-programma's niet kunnen worden geïnterruppeert (gebruik dit met aandacht!).
Interrupties moeten worden ge-activeerd indien deze routine zal worden gebruikt.

CKCNTC 00BD

Uitgevoerd via CALL &H00BD.

Deze routine doet hetzelfde als ISCNTC, maar is iets langzamer. Het is derhalve beter om ISCNTC te gebruiken.

Invoerparameters
Er zijn geen invoerparameters benodigd voor deze routine.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
De accumulator, de flags en de geheugenlocaties INTFLG en KILBUF kunnen worden aangetast door deze routine.

BEEP 0C00

Uitgevoerd via CALL &H00C0.
Deze aanroep veroorzaakt een BEEP-signaal, zodat het lijkt alsof <CTRL> <G> wordt ingedrukt.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL en de geheugenlocaties MUSICF, PLYCNT, VCBA t/m VCBA+4, VCBB t/m VCBB+4 en VCBC t/m VCBC+4 kunnen worden gewijzigd door deze routine.

Opmerkingen
Na het beëindigen van de routine wordt een sprong uitgevoerd naar de BIOS-locatie GICINI, welke de geluidsgenerator terugzet.

CLS 00C3

Uitgevoerd via CALL &H00C3.
Deze aanroep wist het op dat moment geselecteerde scherm (inclusief de grafische schermen).

Invoerparameters
Indien de Z-vlag is opgezet wordt het scherm gewist, anders voert deze aanroep niets uit.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en de geheugenlocaties LINTTB, CSRX en CSRY kunnen

worden gewijzigd door deze aanroep.

POSIT 006C

Uitgevoerd via CALL &H00C6.
Deze routine verplaatst de tekstcursor naar de aangegeven positie.

Invoerparameters
De kolom van de gewenste positie dient in H te staan en de rij in L.

Uitvoerparameters
CSRX en CSRY worden gemodificeerd door deze aanroep.

Gewijzigde registers en geheugenvelden
Het AF-registerpaar en de geheugenvelden CSRX en CSRY worden gewijzigd door deze aanroep.

FNKSB 00C9

Uitgevoerd via CALL &H00C9.
Deze aanroep controleert of het functietoetsoverzicht actief is en drukt de functietoetsen-definities af indien dit het geval is.

Invoerparameters
Indien CNSDFG nullen bevat doet deze routine niets, anders wordt naar DSPFNK overgegaan.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
De registers AF, BC en DE kunnen worden gewijzigd door deze aanroep.

Opmerkingen
Interrupties worden ge-activeerd door deze aanroep.

ERAFNK 00CC

Uitgevoerd via CALL &H00CC.
Deze aanroep onderbreekt het tonen van het functietoetsoverzicht op het beeldscherm, op voorwaarde dat de Video Display Processor in één van de tekstmodi is.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
De registers AF, BC en DE kunnen worden gewijzigd door deze aanroep.

DSPFNK 00CF

Uitgevoerd via CALL &HOOCF.
Deze aanroep zorgt voor het tonen van het functietoetsoverzicht onderaan het beeldscherm.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Geheugenpositie CNSDFG wordt op 255 gezet, daarmee aangevend dat het functietoetsoverzicht actief is.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en de geheugenlocatie CNSDFG kunnen worden gewijzigd door deze aanroep.

Opmerkingen
Interrupties worden ge-activeerd door deze aanroep.

TOTEXT 00D2

Uitgevoerd via CALL &H00D2.
Deze routine forceert het scherm in één van de mogelijke tekstmodi, op voorwaarde dat deze nog niet aanwezig is.

Invoerparameters
De gekozen tekstmodus wordt opgeslagen in de geheugenlocatie OLDSCR.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
Deze aanroep kan de registers AF, BC, DE en HL en de geheugenvelden LINLEN, NAMBAS, CGPBAS en SCRMOD wijzigen.

Opmerkingen
Indien VDP nog niet in de tekstmodus is, roept deze routine de hook H-TOTE aan met de inhoud van OLDSCR in de accumulator. Na terugkeer van de hook wordt de besturing overgegeven aan de BIOS-routine CHGMOD.
Interrupties worden ge-activeerd tijdens deze routine.

CHGCAP 0132

Uitgevoerd via CALL &H0132.
Deze aanroep bestuurt de status van het <CAPS LOCK>-lampje.

Invoerparameters
Indien de accumulator nullen bevat, wordt het <CAPS LOCK>-lampje uitgeschakeld, anders wordt dit aangezet.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
Alleen de accumulator en de flags worden gewijzigd door deze routine.

SNSMAT 0141

Uitgevoerd via CALL &H0141.
Deze aanroep tast een rij van de toetsenbordmatrix af en geeft de status terug van de toetsen in deze rij.

Invoerparameters
Het rijnummer dat moet worden afgetast, wordt in de accumulator gezet.

Uitvoerparameters
De status van toetsen in de rij wordt in de accumulator geplaatst. Indien één van de toetsen is ingedrukt bevat de corresponderende bit in de accumulator een 0, anders een 1.

Gewijzigde registers en geheugenvelden
Alleen de accumulator en de flags worden gewijzigd door deze routine.

Opmerkingen
Interrupties worden ge-activeerd door deze aanroep.

ISFLIO 014A

Uitgevoerd via CALL &H014A.
Deze aanroep controleert of er sprake is van een I/O-actie (invoer/uitvoer-actie) naar een randapparaat.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Is er geen sprake van een I/O dan bevat de accumulator nullen en wordt de Z-flag opgezet, anders wordt de accumulator op een waarde ongelijk aan nul gezet en de Z-flag gewist.

Gewijzigde registers en geheugenvelden
Alleen de accumulator en de flags worden gewijzigd door deze aanroep.

OUTDLP 014D

Uitgevoerd via CALL &H014D.
Deze aanroep wordt gebruikt door de BASIC-vertolker om een teken naar de printer te schrijven.

Invoerparameters
Het af te drukken teken dient te worden geplaatst in de accumulator.

Uitvoerparameters
Er worden geen parameters teruggegeven, maar als de uitvoering wordt onderbroken dan wordt de besturing overgedragen aan de foutafhandelingsroutine van de BASIC-vertolker en er wordt een 'Device I/O error'-melding afgegeven.

Gewijzigde registers en geheugenvelden
Alleen de flags worden aangetast door deze aanroep.

KILBUF 0156

Uitgevoerd via CALL &H0156.
Deze aanroep wist het toetsenbordgeheugen.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
Alleen het HL-registerpaar wordt gewijzigd door deze aanroep.

BIOS-routines die de joystick-poorten besturen

De hier beschreven routines bieden besturing over joystick-poorten en de mogelijk hieraan te koppelen randapparatuur.

GTSTCK 00D5

Uitgevoerd via CALL &H00D5.
Deze routine geeft de status van de cursortoetsen of van één van de joysticks.

Invoerparameters
De accumulator bevat de joystick-identificatie. Deze is 0 voor de cursortoetsen, 1 voor joystick 1 en 2 voor joystick 2.

Uitvoerparameters
De richting van de geselecteerde joystick (of cursortoetsen) wordt geplaatst in de accumulator. Deze posities zijn:

0	een gecentraliseerde joystick
1	omhoog
2	rechts omhoog
3	rechts
4	rechts omlaag
5	omlaag
6	links omlaag
7	links
8	links omhoog

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL kunnen worden gewijzigd door deze aanroep.

Opmerkingen
Interrupties worden ge-activeerd door deze aanroep.

GTTRIG 00D8

Uitgevoerd via CALL &H00D8.

Deze aanroep geeft de huidige status van de spatiebalk, of één van de joystick-vuurknoppen.

Invoerparameters
De accumulator geeft aan welke trekker dient te worden afgetast, zoals hieronder aangegeven:

0 = spatiebalk
1 = trekker 1A
2 = trekker 2A
3 = trekker 1B
4 = trekker 2B

Uitvoerparameters
Indien de bewuste trekker wordt ingedrukt geeft de accumulator 255 terug, anders 0.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL kunnen worden gewijzigd door deze aanroep.

Opmerkingen
Interrupties worden ge-activeerd door deze routine.

GTPAD 00DB

Uitgevoerd via CALL &H00DB.
Deze aanroep geeft de status van een 'touch pad', die is aangesloten op één van de joystick-poorten.

Invoerparameters
De accumulator dient een getal te bevatten tussen 0 en 7, afhankelijk van de benodigde informatie. Indien A in de reeks 0-3 valt, wordt informatie over 'touch pad' aangesloten op joystick-poort 1 gegeven en gelijksoortige informatie wordt gegeven over een 'touch pad' aangesloten op poort 2 indien A valt tussen 4 en 7. Om alleen te weten te komen of een 'touch pad' wordt aangeraakt, kunt u 1 of 5 gebruiken en A=2 of 6 in geval van de y-coördinaat. Om te weten te komen of een 'touch pad'-schakelaar wordt ingedrukt, dient u 3 of 7 in de accumulator te plaatsen.

Uitvoerparameters
De uitvoerparameter wordt geplaatst in de accumulator en hangt af van de invoerparanmeter op de volgende wijze. In geval van 0 of 4 wordt 0 gegeven indien de relevante 'touch pad' wordt ingedrukt. Zo niet, dan bevat A 255.
Hetzelfde geldt voor de status van een 'touch pad'-schakelaar bij een invoerparameter die respectievelijk 1 of 2 is (5 of 6 voor joystick-poort 2).

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL kunnen worden aangetast door deze routine.

Opmerkingen
Deze aanroep activeert interrupties. Zie het BASIC-sleutelwoord PAD voor meer gegevens.

GTPDL 00DE

Uitgevoerd via CALL &H00DE.
Deze aanroep geeft de status van één van de 12 mogelijke 'paddles' die aangesloten kunnen zijn op de joystick-poorten.

Invoerparameters
Het paddle-nummer dient in de accumulator te worden geplaatst. Dit moet een oneven getal zijn voor joystick-poort 1 en een even getal voor joystick-poort 2.

Uitvoerparameters
Een getal in de reeks 0-255 wordt afgegeven in de accumulator, hiermee de status aangevens van de gekozen paddle.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL kunnen worden aangetast door deze aanroep.

Opmerkingen
Deze routine activeert interrupties.

Sectie 4.5

BIOS-routines die horen bij de cassette-interface

De routines in dit gedeelte worden gebruikt om de cassette-interface en het wegschrijfsysteem te besturen.

TAPION 00E1

Uitgevoerd via CALL &H00E1.
Deze routine schakelt de cassettemotor aan en leest de kopregel of header van de tape.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
De 'carry-flag' wordt opgezet indien de uitvoering vroegtijdig wordt afgebroken.

Gewijzigde registers en geheugenvelden
Deze routine kan de registers AF, BC, DE en HL wijzigen.

Opmerkingen
Deze routine de-activeert de interrupties tijdens het lezen van de cassette.

TAPIN 00E4

Uitgevoerd via CALL &H00E4.
Deze aanroep leest een byte van de cassetterecorder.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
De databyte wordt geplaatst in de accumulator. De 'carry-flag' wordt opgezet indien de uitvoering vroegtijdig wordt afgebroken.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL kunnen worden gewijzigd door deze aanroep.

Opmerkingen
Deze routine de-activeert interrupties tijdens het lezen van cassette.

TAPIOF 00E7

Uitgevoerd via CALL &H00E7
Deze aanroep laat de computer stoppen met het lezen van de tape.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Er worden geen uitvoerparameters teruggegeven.

Gewijzigde registers en geheugenvelden
Er worden geen registers of geheugenvelden gewijzigd.

TAPOON 00EA

Uitgevoerd via CALL &H00EA.
Deze aanroep schakelt de cassettemotor aan en schrijft een kopregelblok naar de cassetterecorder.

Invoerparameters
De accumulator moet nul bevatten indien een korte kopregel is geweest, anders wordt een lange kopregel (header) toegepast.

Uitvoerparameters
De 'carry-flag' wordt gezet indien de uitvoering vroegtijdig wordt afgebroken.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL kunnen worden gewijzigd door deze aanroep.

Opmerkingen
Interrupties worden ge-activeerd door deze aanroep.

TAPOUT 00ED

Uitgevoerd via CALL &H00ED.
Deze aanroep schrijft een byte weg naar de cassetterecorder.

Invoerparameters
De accumulator bevat de byte die moet worden weggeschreven.

Uitvoerparameters
Indien de uitvoering vroegtijdig wordt afgebroken, wordt de 'carry-flag' opgezet.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL kunnen worden aangetast door deze aanroep.

Opmerkingen
Deze aanroep activeert interrupties.

TAPOOF 00F0
Uitvoering via CALL &H00F0.
Deze aanroep laat de computer stoppen met het schrijven naar de cassetterecorder.

Invoerparameters
Er zijn geen invoerparameters benodigd.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
Er worden geen registers of geheugenvelden gewijzigd.

STMOTR 00F3
Uitgevoerd via CALL &H00F3.
Deze aanroep schakelt de motor van de cassetterecorder aan of uit of wijzigt de status van de motor in tegenovergestelde toestand.

Invoerparameters
Indien de accumulator 1 bevat, wordt de cassettemotor aangeschakeld; bij 0 wordt deze uitgeschakeld en indien de accumulator 255 bevat wordt de status van de cassettemotor verwisseld ('flipped') dus indien de motor aan is wordt hij uitgeschakeld en andersom.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
Alleen de accumulator en de flags worden aangetast door deze aanroep.

Sectie 4.6

BIOS-routines die geluid behandelen

De volgende routines worden gebruikt om de programmeerbare geluidsgenerator te besturen.

GICINI 0090

Uitgevoerd via CALL &H0090.
Deze aanroep initialiseert de programmeerbare geluidsgenerator, waarbij de geluidswachtrijen worden geleegd en eventueel gegenereerd geluid wordt uitgeschakeld.

Invoerparameters
Er zijn geen invoerparameters benodigd, maar interrupties worden gedeactiveerd door deze routine.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
Alle registers blijven ongewijzigd, maar de geheugenlocaties MUSICF, PLYCNT, VCBA t/m VCBA+4, VCBB t/m VCBB+4 en VCBC t/m VCBC+4 worden op nul gezet.

WRTPSG 0093

Uitgevoerd via CALL &H0093.
Deze aanroep schrijft een waarde naar één van de registers in de programmeerbare geluidsgenerator.

Invoerparameters
De accumulator moet de waarde bevatten van het betreffende register, welke moet liggen tussen 0 en 13. De gegevens die worden geschreven dienen te staan in E.

Uitvoerparameters
Er worden geen parameters teruggegeven.

Gewijzigde registers en geheugenvelden
Er worden geen registers of geheugenvelden gewijzigd.

Opmerkingen
Interrupties worden gede-activeerd aan het begin van deze routine en geactiveerd na voltooiing.

RDPSG 0096
Uitgevoerd via CALL &H0096.
Deze routine leest de inhoud van één van de registers in de programmeerbare geluidsgenerator.

Invoerparameters
De accumulator moet het nummer bevatten van het te lezen register, dat moet liggen tussen 0 en 13.

Uitvoerparameters
De inhoud van het register wordt in de accumulator geplaatst.

Gewijzigde registers en geheugenvelden
Alleen de accumulator wordt gewijzigd.

Opmerkingen
Interrupties worden gede-activeerd aan het begin van de routine en geactiveerd na voltooiing.

STRTMS 0099
Uitgevoerd via CALL &H0099.
Deze routine start de achtergrondmuziek-taak indien er werk is geplaatst in de wachtrij.

Invoerparameters
Er worden geen parameters overgedragen aan deze routine.

Uitvoerparameters
Na voltooiing bevat de accumulator nul indien de geluidsbuffer leeg is.

Gewijzigde registers en geheugenvelden
De registers AF, HL en de geheugenvelden PLYCNT en MUSICF kunnen worden gewijzigd door deze routine.

Opmerkingen
De achtergrondmuziek-taak begint pas na de 50 Hz-interruptie.

Sectie 4.7

BIOS-routines die horen bij de VDP

De BIOS-routines die in dit gedeelte worden behandeld bieden een volledige besturing van de VDP in de MSX-computer.

DISSCR 0041

Uitgevoerd via CALL &H0041

Deze aanroep de-activeert het scherm en wist daarbij het scherm uit met de randkleur. Alle uitvoer wordt naar het scherm gestuurd maar blijft onzichtbaar totdat ENASCR (&H0044) wordt aangeroepen. Het verwisselen van de modus activeert het scherm ook.

Invoerparameters

Geen.

Uitvoerparameters

Geen.

Gewijzigde registers en geheugenvelden

De registers AF en BC worden door deze aanroep gewijzigd en interrupties worden ge-activeerd.

Mogelijke toepassingen

Deze aanroep heeft nut in BASIC, via de USR-functie, om het scherm uit te wissen en zou kunnen worden gebruikt om een plaatje op het scherm te creëren als het scherm is gede-activeerd, waarna activering van het scherm het plaatje opeens zichtbaar laat worden. Voorbeeld:

```
10 DEF USR0=&H41
20 DEF USR1=&H44
30 CLS
40 REM DE-ACTIVEER SCHERM
50 X=USR0(0)
60 INPUT PW$
70 REM ACTIVEER SCHERM
80 X=USR1(0)
```

ENASCR 0044

Uitgevoerd via CALL &H0044.
Dit adres wordt aangeroepen om het scherm te activeren, nadat het werd gede-activeerd door DISSCR.

Invoerparameters
Geen.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
De registers AF en BC worden door deze routine gewijzigd en interrupties worden gede-activeerd.

Mogelijke toepassingen
Het activeren van het scherm na DISSCR. Zie DISSCR voor een voorbeeld.

WRTVDP 0047

Uitgevoerd via CALL &H0047.
Deze aanroep schrijft naar een register van de Video Display Processor. Voor details over het effect van schrijven naar de VDP kunt u de Geavanceerde Programmeringsgids raadplegen, te weten hoofdstuk 16: Video Display Processor.

Invoerparameters
Het nummer van het VDP-register moet worden opgeslagen in register C en de te schrijven databyte in register B.

Uitvoerparameters
Er worden geen uitvoerparameters teruggegeven, maar geheugenlocatie RG0SAV+C bevat de oorspronkelijke waarde van B.

Gewijzigde registers en geheugenvelden
De registers AF en BC en geheugenlocatie RGxSAV worden gewijzigd door deze aanroep, waarin x het nummer van het te beschrijven register voorstelt. Bijvoorbeeld, indien C=5 en B=0 dan bevat geheugenlocatie RG5SAV nul na voltooiing van deze routine. Ook activeert deze routine interrupties.

Mogelijke toepassingen
Dit is een zeer krachtige routine die besturing van de VDP mogelijk maakt. Deze routine wordt aanbevolen om de VDP-registers te beschrijven, omdat automatisch kopieën worden bijgehouden. Indien een waarde wordt geplaatst in een register, is er geen manier om te zeggen wat die waarde is, omdat het 'write-only'-registers zijn (dus zij kunnen alleen worden beschreven).

RDVDP 013E

Uitgevoerd via CALL &H013E.
Deze routine wordt gebruikt om het statusregister van de VDP uit te lezen.

Invoerparameters
Geen.

Uitvoerparameters
De accumulator bevat een kopie van het VDP-statusregister.

Gewijzigde registers en geheugenvelden
Alleen de accumulator wordt gewijzigd door deze aanroep.

RDVRM 004A

Uitgevoerd via CALL &H004A.
Deze aanroep leest een geheugenlocatie in de Video-RAM, die wordt bestuurd door de VDP.

Invoerparameters
Het HL-registerpaar dient het adres te bevatten binnen de te benaderen Video-RAM.

Uitvoerparameters
Na terugkeer bevat A de inhoud van de geheugenlocatie aangegeven door HL. De routine activeert tevens interrupties.
Opmerking: De status van de flags, na voltooiing van deze routine, geven niet de inhoud van de accumulator weer.

Gewijzigde registers en geheugenvelden
Alleen A en de flags worden gewijzigd door deze routine.

WRTVRM 004D

Uitgevoerd via CALL &H004D.
Deze aanroep schrijft de inhoud van de accumulator naar een locatie in de Video-RAM.

Invoerparameters
Het HL-registerpaar bevat het adres binnen de Video-RAM en A de te schrijven waarde.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
De accumulator en de flags worden gewijzigd door deze routine en interrupties worden ge-activeert.

Mogelijke toepassingen
Het benaderen van de Video-RAM.

SETRD 0050
Uitgevoerd via CALL &H0050.
Deze aanroep initieert de VDP voor een leesopdracht.

Invoerparameters
HL dient het adres te bevatten dat wordt gelezen.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
Het A-register en de flags worden gewijzigd en interrupties worden geactiveert door deze aanroep.

Mogelijke toepassingen
Voor het lezen van de VDP-RAM dient de VDP te worden geïnitialiseerd voor een leesopdracht. Deze routine wordt aangeroepen door RDVRM en LDIRMV, dus het is niet nodig om deze routine te gebruiken, tenzij direct uit de VDP wordt gelezen.

SETWRT 0053
Uitgevoerd via CALL &H0053.
Deze aanroep initieert de VDP voor een schrijfopdracht.

Invoerparameters
Het HL-registerpaar bevat het adres in VDP-RAM waarnaar wordt geschreven.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
De accumulator en de flags worden gewijzigd en interrupties worden geactiveerd door deze routine.

Mogelijke toepassingen
Evenals bij SETRD dient deze routine alleen te worden gebruikt, wanneer de VDP direct moet worden benaderd, omdat de routine wordt aangeroepen door de subroutines WRTVRM, FILVRM en LDIRVM.

FILVRM 0056
Uitgevoerd via CALL &H0056.
Deze aanroep vult een gedeelte van de Video-RAM, bestuurd door de VDP, met een enkele waarde.

Invoerparameters
Het adres van de eerste byte in het te vullen blok dient te staan in HL, de lengte van het blok in BC en de te schrijven waarde in A.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
AF en BC worden gewijzigd en interrupties worden ge-activeerd door de aanroep.

Mogelijke toepassingen
Deze routine is handig om een gedeelte van het beeldscherm van een enkele kleur te voorzien. Hij wordt ook gebruikt door de BASIC-PAINT-opdracht.

LDIRMV 0059

Uitgevoerd via CALL &H0059.
Deze aanroep verplaatst een blok Video-RAM, bestuurd door de VDP, naar het hoofdgeheugen.

Invoerparameters
Het adres in de Video-RAM staat in HL, de lengte van het blok in BC en de bestemming in het hoofdgeheugen staat in DE.

Uitvoerparameters
Een blok geheugen zoals hierboven vermeld met DE en BC wordt teruggegeven door de routine.

Gewijzigde registers en geheugenvelden
AF, BC, DE en het bestemmingsblok worden gewijzigd door deze routine.

Mogelijke toepassingen
Deze aanroep is handig indien veel verwerking op het beeldscherm nodig is, omdat het veel sneller is het scherm te laden in het geheugen, het te verwerken en het vervolgens terug te plaatsen in de VDP-RAM met LDIRVM.

Opmerkingen
Verwar deze routine niet met LDIRVM, LDIRMV verplaatst een blok vanuit de VDP-RAM.

LDIRVM 005C

Uitgevoerd via CALL &H005C.
Deze routine kopieert een blok geheugen in de RAM die wordt bestuurd door de VDP.

Invoerparameters
Het adres van het blok dient in HL te staan, de bestemming in de Video-RAM

in DE en de lengte in BC.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
AF, BC en DE worden gewijzigd en interrupties worden ge-activeerd door deze aanroep.

Opmerkingen
Verwar deze routine niet met LDIRMV, LDIRVM verplaatst een blok naar de Video-RAM.

CHGMOD 005F
Uitgevoerd via CALL &H005F.
Deze aanroep zet de VDP in één van de vier mogelijke bewerkingsmodi, dit zijn de tekstmodus, de grafische modus I (32 kolommen) en de grafische modus II of meerkleurenmodus.

Invoerparameters
De accumulator moet de betreffende VDP-modus bevatten en wel als volgt:

A=0 -> tekstmodus (40 kolommen)
A=1 -> grafische modus I (32 kolommen)
A=2 -> grafische modus II (HIRES)
a=3 -> meerkleurenmodus

Uitvoerparameters
Na voltooiing bevat SCRMOD het modusnummer.

Gewijzigde registers en geheugenvelden
De gewijzigde registers zijn AF, BC, DE en HL. De geheugenlocaties die mogelijkerwijs kunnen worden gewijzigd door deze aanroep zijn LINLEN, NAMBAS, CGPBAS, SCRMOD en OLDSCR.

Opmerkingen
Deze routine roept één van de routines INITXT, INIT32, INIGRP of INIMULT aan, afhankelijk van de waarde in de accumulator. Na voltooiing worden interrupties ge-activeerd.

CHGCLR 0062
Uitgevoerd via CALL &H0062.
Deze aanroep wijzigt de voorgrondkleur, de achtergrondkleur en de randkleur indien de VDP in de tekstmodus is (40 of 32 kolommen); of wijzigt de randkleur indien de grafische modus actief is.

Invoerparameters
De voorgrondkleur wordt bepaald vanuit FORCLR. De achtergrondkleur wordt bepaald vanuit BAKCLR. De randkleur wordt bepaald vanuit BDRCLR.

Uitvoerparameters
Geen

Gewijzigde registers en geheugenvelden
AF, BC en HL worden gewijzigd door deze aanroep.

CLRSPR 0069

Uitgevoerd via CALL &H0069.
Deze aanroep initialiseert alle sprites. De patronen worden transparant gemaakt (nullen), de sprite-kleuren worden gelijkgesteld aan de huidige voorgrondkleur; de verticale posities worden op 209 gezet (van het scherm); de sprite-namen worden gekoppeld aan het sprite-vlaknummer (dus de naam van de sprite in vlak 0 wordt gekoppeld aan ASCII-code 0 enz.).

Invoerparameters
Geen.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE en HL worden gewijzigd door deze aanroep.

Opmerkingen
Deze routine wordt niet aangeroepen bij het wijzigen van een modus.

INITXT 006C

Uitgevoerd via CALL &H006C.
Deze aanroep initialiseert de geheugenlocaties die het scherm beschrijven. In geval van de tekstmodus wordt SETTXT aangeroepen.

Invoerparameters
De geheugenlocaties TXTNAM, TXTCGP en LINL40 bevatten de invoerparameters.

Uitvoerparameters
Na voltooiing bevat SCRMOD=0, OLDSCR=0, NAMBAS=TXTNAM, CGPBAS=TXTCGP en LINLEN=LINL40.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE, HL, geheugenvelden RG0SAV, RG1SAV, RG2SAV, RG3SAV, RG4SAV, RG5SAV, RG6SAV en de geheugenvelden aangegeven als uitvoerparameters, worden gewijzigd door deze aanroep.

Opmerkingen
De interrupties worden ge-activeerd door deze routine.

INIT32 006F

Uitgevoerd via CALL &H006F.
Deze aanroep initialiseert de geheugenvelden welke het scherm voor de grafische modus I (32 kolommen tekst) beschrijven en roept vervolgens SETT32 aan.

Invoerparameters
De geheugenvelden T32NAM, T32CGP, T32COL, T32ATR, T32PAT en LINL32 bevatten de invoerparameters voor deze routine.

Uitvoerparameters
Na voltooiing geldt: SCRMOD=1, OLDSCR=1, NAMBAS=T32NAM, CGPBAS=CGPBASS, PATBAS=T32BAS, ATRBAS=T32ATR en LINLEN=LINL32.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE, HL, de geheugenvelden RG0SAV, RG1SAV, RG2SAV, RG3SAV, RG4SAV, RG5SAV, RG6SAV en de geheugenvelden genoemd als uitvoerparameters worden gewijzigd door deze routine.

Opmerkingen
Interrupties worden ge-activeerd door deze routine.

INIGRP 0072

Uitgevoerd via CALL &H0072.
Deze aanroep initialiseert de verschillende geheugenlocaties voor de grafische modus II (HIRES-modus) en roept vervolgens SETGRP aan.

Invoerparameters
De geheugenvelden GRPNAM, GRPCGP, GRPCOL, GRPATR en GRPPAT bevatten de invoerparameters voor deze routine.

Uitvoerparameters
Na voltooiing geldt: SCRMOD=2, PATBAS=GRPPAT en ATRBAS=GRPATR.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE, HL, de geheugenvelden RG0SAV, RG1SAV, RG2SAV, RG3SAV, RG4SAV, RG5SAV, RG6SAV en de geheugenvelden genoemd als uitvoerparameters worden gewijzigd door deze routine.

Opmerkingen
Interrupties worden ge-activeerd door deze routine.

INIMLT 0075

Uitgevoerd via CALL &H0075.

Deze aanroep initialiseert de verschillende geheugenvelden voor de meerkleurenmodus en roept vervolgens SETMLT aan.

Invoerparameters
De geheugenvelden MLTNAM, MLTCOL, MLTATR en MLTPAT bevatten de invoerparameters.

Uitvoerparameters
Na voltooiing geldt: SCRMOD=3, PATBAS=MLTPAT en ATRBAS=MLTATR.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE, HL, de geheugenvelden RG0SAV, RG1SAV, RG2SAV, RG3SAV, RG4SAV, RG5SAV, RG6SAV en de geheugenvelden genoemd als uitvoerparameters worden gewijzigd door deze routine.

Opmerkingen
Deze routine activeert interrupties.

SETTXT 0078

Uitgevoerd via CALL &H0078.
Deze aanroep initialiseert de VDP-registers voor de tekstmodus (40 kolommen).

Invoerparameters
Geen.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE, HL, de geheugenvelden RG0SAV, RG1SAV, RG2SAV, RG3SAV, RG4SAV, RG5SAV, RG6SAV worden gewijzigd door deze routine.

Opmerkingen
Het aanroepen van deze routine is niet voldoende om van modus te veranderen. De VDP-registers worden weliswaar gezet, maar de Video-RAM wordt niet geïnitialiseerd. Interrupties worden ge-activeerd gedurende de uitvoering van deze routine.

SETT32 007B

Uitgevoerd via CALL &H007B.
Deze aanroep intialiseert de VDP-registers voor de grafische modus I (32 kolommen).

Invoerparameters
Geen.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE, HL, de geheugenvelden RG0SAV, RG1SAV, RG2SAV, RG3SAV, RG4SAV, RG5SAV, RG6SAV worden gewijzigd door deze routine.

Opmerkingen
Het aanroepen van deze routine is niet voldoende om van modus te veranderen. De VDP-registers worden weliswaar gezet, maar de Video-RAM wordt niet geïnitialiseerd. Interrupties worden ge-activeerd gedurende de uitvoering van deze routine.

SETGRP 007E
Uitgevoerd via CALL &H007E.
Deze aanroep initialiseert de VDP-registers voor de grafische modus II (HIRES).

Invoerparameters
Geen.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE, HL, de geheugenvelden RG0SAV, RG1SAV, RG2SAV, RG3SAV, RG4SAV, RG5SAV, RG6SAV worden gewijzigd door deze routine.

Opmerkingen
Het aanroepen van deze routine is niet voldoende om van modus te veranderen. De VDP-registers worden weliswaar gezet, maar de Video-RAM wordt niet geïnitialiseerd. Interrupties worden ge-activeerd gedurende de uitvoering van deze routine.

SETMLT 0081
Uitgevoerd via CALL &H0081.
Deze aanroep initialiseert de VDP-registers voor de meerkleurenmodus.

Invoerparameters
Geen.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
De registers AF, BC, DE, HL, de geheugenvelden RG0SAV, RG1SAV, RG2SAV, RG3SAV, RG4SAV, RG5SAV, RG6SAV worden gewijzigd door deze routine.

Opmerkingen
Het aanroepen van deze routine is niet voldoende om van modus te veranderen. De VDP-registers worden weliswaar gezet, maar de Video-RAM wordt niet geïnitialiseerd. Interrupties worden ge-activeerd gedurende de uitvoering van deze routine.

CALPAT 0084
Uitgevoerd via CALL &H0084.
Deze aanroep geeft het adres van een sprite-patroon in de video-RAM.

Invoerparameters
Het sprite-nummer moet in de accumulator staan.

Uitvoerparameters
Het adres van de sprite in de video-RAM wordt geplaatst in het HL-registerpaar.

Gewijzigde registers en geheugenvelden
De registers AF, DE en HL worden gewijzigd door deze aanroep.

Opmerkingen
De status van de interruptie-flip-flop wordt door deze aanroep niet gewijzigd.

CALATR 0087
Uitgevoerd via CALL &H0087.
Deze routine geeft het adres in de video-RAM van de attributentabel van een sprite.

Invoerparameters
Het sprite-nummer moet in de accumulator staan.

Uitvoerparameters
Na terugkeer bevat het HL-registerpaar het adres van de attributentabel in de video-RAM van de betreffende sprite.

Gewijzigde registers en geheugenvelden
DE, HL en de flags worden gewijzigd door deze routine.

Opmerkingen
Interrupties worden niet beïnvloed.

GSPSIZ 008A
Uitgevoerd via CALL &H008A.
Deze routine geeft de grootte terug van een sprite, uitgedrukt in het aantal benodigde bytes (8 of 32).

Invoerparameters
Geen.

Uitvoerparameters
Het aantal bytes per sprite wordt in de accumulator geplaatst.

Gewijzigde registers en geheugenvelden
De accumulator en de flags worden gewijzigd door deze routine.

Opmerkingen
Indien 16x16-sprites in gebruik zijn wordt de 'carry-flag' gezet na terugkeer, anders wordt deze gewist. Interrupties worden niet beïnvloed.

GRPPRT 008D

Uitgevoerd via CALL &H008D.
Deze aanroep wordt gebruikt om een teken op het grafische scherm (HIRES-modus) af te drukken op de plaats van de grafische cursor.

Invoerparameters
De code van het af te drukken teken dient in de accumulator te worden geplaatst.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
Er worden geen registers of geheugenvelden gewijzigd door deze routine.

Opmerkingen
Deze routine werkt alleen in de grafische modus II.

SCALXY 010E

Uitgevoerd via CALL &H010E.
Deze aanroep verzekert dat een met registers aangegeven punt zich op het scherm bevindt. Zoniet, dan worden de aangegeven coördinaten afgekapt en een foutindicator opgezet. Het afkappen betekent dat indien x of y te groot blijken te zijn, de maximum toegestane waarden hieraan worden toegekend. Blijken de waarden negatief te zijn dan worden ze vervangen door 0. In geval van meerkleurenmodus worden x en y gedeeld door 4 omdat deze modus 64 horizontale cellen heeft en 48 verticale cellen.

Invoerparameters
BC moet de x-coördinaat bevatten en DE de y-coördinaat.

Uitvoerparameters
Na terugkeer bevatten BC en DE de afgekapte x- en y-coördinaten. Blijkt x of y buiten de toegestane waarden te vallen dan wordt de 'carry-flag'

gewist, anders wordt deze gezet.

Gewijzigde registers en geheugenvelden
AF, BC en DE kunnen worden gewijzigd door deze routine.

Opmerkingen
Let er op dat x en y geschaald zijn indien de meerkleurenmodus actief is (pas de schaling niet zelf nog eens toe!).
De waarden van x en y zijn achtereenvolgens 0-255 en 0-191 in de grafische modus II of meerkleurenmodus.

MAPXYC 0111

Uitgevoerd via CALL &H0111.
Deze zeer handige routine berekent het adres in de video-RAM van een pixel in zowel de grafische modus II als de meerkleurenmodus. Ook wordt de positie van de bits in de byte aangegeven die de betreffende pixel voorstellen.

Invoerparameters
De positie (x,y) van het betreffende punt dient te worden aangegeven in BC(x) en DE(y). Voor de grafische modus II moet x liggen tussen 0-255 en y tussen 0-191. In de meerkleurenmodus moet x liggen tussen 0-63 en y tussen 0-47. Deze routine gebruikt SCRMOD om de schermmodus te bepalen, GRPCGP om het beginadres van de patroongeneratortabel in de grafische modus II en MLTCGP om het beginadres van de patroongenerator in de meerkleurenmodus te bepalen.

Uitvoerparameters
Het adres in de video-RAM van de byte die de pixel bevat wordt geplaatst in CLOC en in CMASK de byte die aangeeft welke de relevante bits zijn die de pixel beschrijven. In de grafische modus II wordt de betreffende pixel aangegeven met één bit (voorgrond=1, achtergrond=0). Deze bit staat op dezelfde positie als die in CMASK, bijvoorbeeld indien de pixel is opgeslagen in bit 5 van de byte, aangegeven door CLOC, dan bevat CMASK 00100000 binair. Voor de meerkleurenmodus wordt iedere pixel voorgesteld door vier bits - de kleur van de pixel en de positie van de vier bits corresponderen ook hier met de bits in CMASK.
Bijvoorbeeld indien de pixel staat opgeslagen in de bits 0-3 van de byte aangegeven door CLOC, dan bevat CMASK 00001111 binair.

Gewijzigde registers en geheugenvelden
De registers AF, D, HL en de geheugenvelden CLOC en CMASK worden gewijzigd door deze aanroep.

Opmerkingen
Een controle of de pixel zich op het scherm bevindt wordt niet uitgevoerd. Om deze controle uit te voeren en de schaling toe te passen op de x- en y-waarden in de meerkleurenmodus, kan de subroutine SCALXY worden gebruikt.

FETCHC 0114

Uitgevoerd via CALL &H0114.
Deze routine leest eenvoudigweg de huidige pixelpositie en maskerpatroon.

Invoerparameters
CLOC dient het huidige pixeladres te bevatten, CMASK het maskerpatroon.

Uitvoerparameters
Na voltooiing bevat HL de inhoud van CLOC en A de waarde die staat opgeslagen in CMASK.

Gewijzigde registers en geheugenvelden
HL en AF zijn de enige registers die worden gewijzigd door deze aanroep.

Opmerkingen
Deze aanroep kan worden vervangen door twee Z80-instructies, namelijk:

```
LD A,(CMASK)
LD HL,(CLOC)
```

STOREC 0117

Uitgevoerd via CALL &H0117.
Deze routine slaat eenvoudigweg het huidige pixeladres en maskerpatroon op in het geheugen.

Invoerparameters
HL dient het pixeladres te bevatten en A het maskerpatroon.

Uitvoerparameters
Na voltooiing bevat CMASK een kopie van de accumulator, CLOC bevat de waarde van HL.

Gewijzigde registers en geheugenvelden
De geheugenvelden CMAKS en CLOC worden gewijzigd door deze aanroep.

Opmerkingen
Deze aanroep kan door de volgende Z80-instructies worden vervangen, namelijk:

```
LD (CMASK),A
LD (CLOC),HL
```

SETATR 011A

Uitgevoerd via CALL &H011A.
Deze aanroep vervangt de inhoud van de attributenbyte door die van de accumulator indien de waarde hiervan minder dan 16 is, anders blijft de attributenbyte ongewijzigd en wordt een foutindicator opgezet.

Invoerparameters
De accumulator moet de waarde bevatten die in de attributenbyte moet worden geplaatst.

Uitvoerparameters
Na voltooiing bevat ATRBYT (de attributenbyte) de waarde meegegeven in de accumulator, op voorwaarde dat de waarde minder is dan 16.

Gewijzigde registers en geheugenvelden
ATRBYT kan worden gewijzigd door deze aanroep.

Opmerkingen
De attributenbyte geeft de kleur van een pixel aan. Deze wordt gebruikt door SETC om de pixel op deze kleur te zetten.

READC 011D

Uitgevoerd via CALL &H011D.
Deze aanroep leest de attributen (kleur) van de huidige pixel.

Invoerparameters
Het adres van de huidige pixel en het maskerpatroon dienen achtereenvolgens te worden opgeslagen in CLOC en CMASK.

Uitvoerparameters
Na voltooiing wordt de kleur van de huidige pixel opgeslagen in de accumulator.

Gewijzigde registers en geheugenvelden
Alleen de accumulator en de flags worden beïnvloed door deze aanroep.

SETC 0120

Uitgevoerd via &H0120.
Deze aanroep bepaalt de kleur van de huidige pixel. In geval van meerkleurenmodus is dit het enige effect op het beeldscherm. Echter, als in de grafische modus II de pixelkleur niet gelijk is aan de huidige voorgrond- of achtergrondkleur, dan wordt de voorgrondkleur van alle pixels in de byte die de betreffende pixel bevat, gewijzigd in de aangegeven kleur.

Invoerparameters
Het huidige pixeladres en maskerpatroon dienen te worden geplaatst in CLOC en CMASK. De kleur waarin de pixel moet worden aangepast dient te worden gespecificeerd in ATRBYT.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
Alleen de accumulator en de flags worden gewijzigd door deze routine.

Opmerkingen
Indien alle pixels in een bepaalde byte in dezelfde kleur worden gewijzigd dan wordt deze kleur de nieuwe achtergrondkleur: de byte in video-RAM aangegeven door CLOC zal op nullen staan.

RIGHTC 00FC

Uitgevoerd via CALL &H00FC.
Deze routine verplaatst de grafische cursor één pixel naar rechts.

Invoerparameters
Het adres van de byte met de betreffende pixel wordt opgeslagen in CLOC en het pixel-maskerpatroon in CMASK.

Uitvoerparameters
De inhoud van CLOC en CMASK wordt bijgewerkt, zodat ze de nieuwe positie van de grafische cursor bevatten.

Gewijzigde registers en geheugenvelden
AF en de geheugenvelden CLOC en CMASK kunnen worden gewijzigd door deze routine.

Opmerkingen
Wordt de grafische cursor rechts van het scherm af bewogen dan verschijnt deze vervolgens aan de linkerkant, één pixelpositie lager. Er wordt geen controle uitgevoerd op het van het scherm af bewegen, rechts onderaan het scherm.
In de meerkleurenmodus wordt de grafische cursor verplaatst met een blok van 4x4 pixels.

LEFTC 00FF

Uitgevoerd via CALL &H00FF.
Deze routine verplaatst de grafische cursor één pixel naar links.

Invoerparameters
Het adres van de byte met de betreffende pixel wordt opgeslagen in CLOC en het pixel-maskerpatroon in CMASK.

Uitvoerparameters
De inhouden van CLOC en MASK worden bijgewerkt, zodat ze de nieuwe positie van de grafische cursor bevatten.

Gewijzigde registers en geheugenvelden
AF en de geheugenvelden CLOC en CMASK kunnen worden gewijzigd door deze routine.

Opmerkingen
Wordt de grafische cursor links van het scherm af bewogen dan verschijnt deze vervolgens aan de rechterkant, één pixelpositie hoger. Er wordt geen controle uitgevoerd op het van het scherm af bewegen, links bovenaan het scherm.
In de meerkleurenmodus wordt de grafische cursor verplaatst met een blok van 4x4 pixels.

UPC 0102

Uitgevoerd via CALL &H0102.
Deze routine verplaatst de grafische cursor één pixel omhoog, indien de top van het scherm nog niet is bereikt., anders blijft de positie ongewijzigd.

Invoerparameters
Het adres van de byte met de betreffende pixel wordt opgeslagen in CLOC en het pixel-maskerpatroon in CMASK.

Uitvoerparameters
De inhouden van CLOC en CMASK worden bijgewerkt, zodat ze de nieuwe positie van de grafische cursor bevatten.

Gewijzigde registers en geheugenvelden
AF en de geheugenvelden CLOC en CMASK kunnen worden gewijzigd door deze routine.

Opmerkingen
In de meerkleurenmodus wordt de grafische cursor verplaatst met een blok van 4x4 pixels.

TUPC 0105

Uitgevoerd via CALL &H00FC.
Deze routine verplaatst de grafische cursor één pixel omhoog indien de top van het scherm nog niet is bereikt, anders blijft de positie ongewijzigd en wordt de 'carry-flag' opgezet.

Invoerparameters
Het adres van de byte met de betreffende pixel wordt opgeslagen in CLOC en het pixel-maskerpatroon in CMASK.

Uitvoerparameters
De inhouden van CLOC en CMASK worden bijgewerkt, zodat ze de nieuwe positie van de grafische cursor bevatten.

Gewijzigde registers en geheugenvelden
AF en de geheugenvelden CLOC en CMASK kunnen worden gewijzigd door deze routine.

Opmerkingen
Deze aanroep is vrijwel gelijk aan UPC; het enige verschil is dat de 'carry-flag' wordt opgezet indien de grafische cursor de top van het beeldscherm heeft bereikt, terwijl deze anders wordt gewist.
In de meerkleurenmodus wordt de grafische cursor verplaatst met een blok van 4x4 pixels.

DOWNC 0108

Uitgevoerd via CALL &H0108.
Deze routine verplaatst de grafische cursor één pixel omlaag, indien deze de onderkant van het scherm nog niet heeft bereikt, anders blijft de positie ongewijzigd.

Invoerparameters
Het adres van de byte die de pixel bevat wordt opgeslagen in CLOC en het pixel-maskerpatroon in CMASK.

Uitvoerparameters
De inhouden van CLOC en CMASK zijn bijgewerkt voor de nieuwe positie van de cursor.

Gewijzigde registers en geheugenvelden
AF en de geheugenvelden CLOC en CMASK kunnen worden gewijzigd door deze aanroep.

Opmerkingen
In de meerkleurenmodus wordt de grafische cursor verplaatst over een blok van 4x4 pixels.

TDOWNC 010B

Uitgevoerd via CALL &H010B.
Deze routine verplaatst de grafische cursor één pixel omlaag, indien deze de onderkant van het scherm nog niet heeft bereikt, anders blijft de positie ongewijzigd en wordt de 'carry-flag' gezet.

Invoerparameters
Het adres van de byte die de pixel bevat wordt opgeslagen in CLOC en het pixel-maskerpatroon in CMASK.

Uitvoerparameters
De inhouden van CLOC en CMASK worden bijgewerkt voor de nieuwe positie van de grafische cursor.

Gewijzigde registers en geheugenvelden
AF en de geheugenlocaties CLOC en CMASK kunnen worden gewijzigd door deze routine.

Opmerkingen
Deze aanroep is vrijwel gelijk aan DOWNC; het enige verschil is dat de 'carry-flag' wordt opgezet indien de grafische cursor de onderkant van het scherm reeds heeft bereikt en anders wordt de 'carry-flag' gewist.
In de meerkleurenmodus wordt de grafische cursor verplaatst met blokken van 4x4 pixels.

NSETCX 0123

Uitgevoerd via CALL &H0123.
Deze aanroep zet de kleur van een opgegeven aantal pixels rechts van de huidige positie van de grafische cursor op een meegegeven kleur.

Invoerparameters
HL moet het aantal te wijzigen pixels bevatten, ATRBYT moet de kleur van de pixels bevatten. De positie van de grafische cursor en het maskerpatroon moeten worden opgeslagen in respectievelijk CLOC en CMASK.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
AF, BC, DE, HL en de geheugenvelden CLOC en CMASK kunnen worden beïnvloed door deze aanroep.

Opmerkingen
In de meerkleurenmodus verplaatst de cursor een pixel naar links, maar in de grafische modus II blijft deze positie ongewijzigd.
Interrupties worden ge-activeerd door deze routine.

GTASPC 0126

Uitgevoerd via CALL &H0126.
Deze routine wordt gebruikt door de BASIC-vertolker, namelijk bij het CIRCLE-commando, om de huidige aspect-ratioparameters op te halen.

Invoerparameters
ASPCT1 moet bevatten: 256/(aspect-ratio). ASPCT2 moet bevatten: 256*(aspect-ratio).

Uitvoerparameters
Na voltooiing moet HL de inhoud van ASPCT2 en DE die van ASPCT1 bevatten.

Gewijzigde registers en geheugenvelden
Alleen HL en DE worden door deze routine gewijzigd.

PNTINI 0129

Uitgevoerd via CALL &H0129.

Dit is een initialisatieroutine voor de BASIC PAINT-opdracht. De randkleur van de te kleuren vorm wordt gecontroleerd of deze minder is dan 16 en vervolgens bewaard.

Invoerparameters
De accumulator moet de kleur van de rand bevatten die moet worden gekleurd.

Uitvoerparameters
Indien de meerkleurenmodus actief is, bevat BRDATR een kopie van de invoerparameters, anders bevat deze een kopie van de waarde die in ATRBYT staat. Ook indien de invoerparameter groter is dan 15 in de meerkleurenmodus wordt de 'carry-flag' opgezet na voltooiing.

Gewijzigde registers en geheugenvelden
De accumulator, de flags en de geheugenlocatie BRDATR kunnen worden gewijzigd door deze routine.

Opmerkingen
De randkleur wordt alleen gebruikt in de meerkleurenmodus-PAINT-routine. In de grafische modus II zoekt PAINT naar de huidige voorgrondkleur.

SCANR 012C

Uitgevoerd via CALL &H012C.
Deze aanroep tast een opgegeven aantal pixels af rechts van de grafische cursor, waarbij deze pixels op een meegegeven kleur worden gezet totdat een bepaalde randkleur-pixel wordt gevonden of de rechterzijde van het beeldscherm is bereikt of het maximum aantal af te tasten pixels is bereikt.

Invoerparameters
Het maximum aantal af te tasten pixels dient te worden opgeslagen in DE (tot 256), de randkleur moet staan in BRDATR en de kleur waarin de pixels moeten worden gewijzigd moet staan in ATRBYT.

Uitvoerparameters
Geen.

Gewijzigde registers en geheugenvelden
AF, BC, DE, HL en de geheugenvelden CLOC, CMASK en FILNAM t/m FILNAM+3 kunnen worden gewijzigd door deze aanroep.

Opmerkingen
Deze routine wordt gebruikt door de PAINT-instructie; FILNAM t/m FILNAM+3 worden als werkgebied gebruikt.
Deze aanroep activeert interrupties.

SCANL 012F

Uitgevoerd via CALL &H012F.

Deze aanroep tast een opgegeven aantal pixels af links van de grafische cursor, waarbij deze pixels op een meegegeven kleur worden gezet totdat een bepaalde randkleur-pixel wordt gevonden of de linkerzijde van het beeldscherm is bereikt of het maximum aantal af te tasten pixels is bereikt.

Invoerparameters

Het maximum aantal af te tasten pixels dient te worden opgeslagen in DE (tot 256), de randkleur moet staan in BRDATR en de kleur waarin de pixels moeten worden gewijzigd, moet staan in ATRBYT.

Uitvoerparameters

Geen.

Gewijzigde registers en geheugenvelden

AF, BC, DE, HL en de geheugenvelden CLOC, CMASK en FILNAM t/m FILNAM+3 kunnen worden gewijzigd door deze aanroep.

Opmerkingen

Deze routine wordt gebruikt door de PAINT-instructie; FILNAM t/m FILNAM+3 worden als werkgebied gebruikt.

Deze aanroep activeert interrupties.

Sectie 4.8

Het gebruik van 'hooks'

Hooks bieden een mogelijkheid om de BASIC-vertolker op bepaalde punten te onderscheppen, om zodoende additionele of alternatieve processen uit te voeren. Om deze laatste regel toe te lichten is het het eenvoudigst om een voorbeeld te gebruiken. Beschouw bijvoorbeeld de BIOS-aanroep CHPUT (zie sectie 2.3). Deze routine wordt gebruikt om tekst op het scherm af te drukken. Hier volgt een dis-assembly van een eerste paar instructies van CHPUT:

```
PUSH HL          : BEWAAR ALLE TE GEBRUIKEN REGS
PUSH DE
PUSH BC
PUSH AF
CALL &HFDA4      : ROEP HOOK H.CHPU AAN
```

Allereerst worden de registers HL, BC, DE en AF op de stack geplaatst. Daarna wordt geheugenadres &HFDA4 aangeroepen. Gewoonlijk is de inhoud van &HFDA4 en de daaropvolgende vier bytes een Z80-instructie, die direct besturing overgeeft aan de instructie die volgt op de CALL. Echter, &HFDA4 staat in RAM, zodat zijn inhoud kan worden gewijzigd. Laat de drie bytes veranderen, te beginnen bij &HFDA4, opdat de besturing wordt overgedragen aan een andere subroutine op adres &HD000. Dus:

```
FDA4  C3  00  00  JP   &HD000
```

Laat de codering als volgt zijn:

```
POP HL           : VERWIJDER EERSTE ITEM VAN STACK
                   (DUS GEEFT HOOK-ADRES TERUG)
POP AF           : PLAATS REGS TERUG
POP BC
POP DE
POP HL
RET              : KEER TERUG UIT BIOS CALL CHPUT
```

Nu is het item op de stack 3 meer dan het adres waarvan &HFDA4 werd aangeroepen, dit is het terugkeeradres naar CHPUT. Daarom, indien dit wordt

verwijderd van de stack, wordt de stack teruggebracht in de staat waarin deze was direct voor de aanroep van &HFDA4. Ook indien de volgende vier items worden gePOPt zoals hierboven getoond, zal de stack worden zoals hij was na het aanroepen van CHPUT, dus RET zal de subroutine CHPUT beëindigen. Als gevolg hiervan worden de tekens niet afgedrukt die zouden moeten worden afgedrukt!
Dit voorbeeld demonstreert hoe de vertolker kan worden onderschept. De vijf bytes vanaf &HFDA4 staan bekend als een hook. Gelijksoortige 'hooks' bestaan op verschillende plaatsen in het besturingssysteem en de BASIC-vertolker.
Een lijst ervan, op alfabetische volgorde, vindt u hierna.

NAAM	ADRES	BESCHRIJVING
H-ATTR	FE1C	MSXSTS, aan het begin van de ATTR$. (attributen) routine.
H-BAKU	FEAD	SPCDSK, bij de BAKUPT (BACK UP)-routine.
H-BINL	FE76	SPCDSK, bij de BINLOD (BINAIRE LOAD)-routine.
H-BINS	FE71	SPCDSK, bij de BINSAV (BINAIRE SAVE)-routine.
H-BUFL	FF8E	BINTRP, bij de BUFLIN (BUFFERREGEL)-routine.
H-CHGE	FDC2	MSXIO, aan het begin van de CGET-routine.
H-CHPU	FDA4	MSXIO, aan het begin van de CHPUT-routine.
H-CHRG	FF48	BINTRP.
H-CLEA	FED0	BIMISC, bij de CLEARC-routine.
H-CMD	FE0D	MSXSTS, aan het begin van de CMD (COMMANDO)-routine.
H-COMP	FF57	BINTRP.
H-COPY	FE08	MSXSTS, aan het begin van de COPY-routine.
H-CRDO	FEE9	BIO, bij de CRDO (CRIF DO)-routine.
H-CRUN	FF20	BINTRP.
H-CRUS	FF25	BINTRP.
H-CVD	FE49	MSXSTS, aan het begin van de CVD-routine.
H-CVI	FE3F	MSXSTS, aan het begin van de CVI-routine.
H-CVS	FE44	MSXSTS, aan het begin van de CVS-routine.
H-DEVN	FEC1	SPCDEV, bij de DEVNAM-routine.
H-DGET	FE80	SPCDSK, bij de DGET-routine.
H-DIRD	FF11	BINTRP, bij de DIRDO-aanroep.
H-DOGR	FEF3	GENGRP, bij de DOGRPH-routine.
H-DSKC	FEEE	BIO, bij de DSKCHI-routine.
H-DSKF	FE12	MSXSTS, aan het begin van de DSKF-routine.
H-DSKI	FE17	MSXSTS, aan het begin van de DSKI-routine.
H-DSKO	FDEF	MSXSTS, aan het begin van de DSKO$-routine.
H-DSPC	FDA9	MSXIO, aan het begin van de DSPCSR-routine.
H-DSPF	FDB3	MSXIO, aan het begin van de DSPFNK-routine.
H-EOF	FEA3	SPCDSK, bij de EOF-functie.
H-ERAC	FDAE	MSXIO, bij het begin van de ERACSR-routine.
H-ERAF	FDB8	MSXIO, bij het begin van de ERAFNK-routine.
H-ERRF	FF02	BINTRP.
H-ERRO	FFB1	BINTRP, bij de ERROR-routine.

H-ERRP	FEFD	BINTRP, bij de ERRPRT-routine.
H-EVAL	FF70	BINTRP.
H-FIEL	FE2B	MSXSTS, aan het begin van de FIELD-routine.
H-FILE	FE7B	SPCDSK, bij het FILES-commando.
H-FILO	FE85	SPCDSK, bij de FILOU1-routine.
H-FINE	FF1B	BINTRP.
H-FING	FF7A	BINTRP.
H-FINI	FF16	BINTRP.
H-FINP	FF5C	BINTRP.
H-FORM	FFAC	MSXIO, bij de FORMAT-routine.
H-FPOS	FEA8	SPCDSK, bij de FPOS-functie.
H-FRET	FF9D	BISTRG, bij de FRETM-routine.
H-FRME	FF66	BINTRP.
H-FRQI	FF93	BINTRP, bij de FRQINT-routine.
H-GEND	FEC6	SPCDEV, bij de GENDSP-routine.
H-GETP	FE4E	SPCDSK, bij de GETPTR-routine.
H-GONE	FF43	BINTRP.
H-INDS	FE8A	SPCDSK, bij de INDSKC-routine.
H-INIP	FDC7	MSXIO, bij het begin van INIPAT.
H-INLI	FDE5	MSXINL, aan het begin van de LININ-routine.
H-IPL	FE03	MSXSTS, aan het begin van de IPL-routine.
H-ISFL	FEDF	BIMISC, bij de ISFLIO-routine.
H-ISMI	FF7F	BINTRP, bij de ISMID$ (is MID$)-routine.
H-ISRE	FF2A	BINTRP.
H-KEYC	FDCC	MSXIO, aan het begin van KEYCOD.
H-KEYI	FD9A	MSXIO, aan het begin van de INTERRUPTIE-routine.
H-KILL	FDFE	MSXSTS, aan het begin van de KILL-routine.
H-KYEA	FDD1	MSXIO, aan het begin van de KYEASY-routine.
H-LIST	FF89	BINTRP, bij de LIST-routine.
H-LOC	FE99	SPCDSK, bij de LOC-functie.
H-LOF	FE9E	SPCDSK, bij de LOF-functie.
H-LOPD	FED5	BIMISC, bij de LOPDFT-routine.
H-LPTO	FFB6	MSXIO, bij de LPTOUT-routine.
H-LPTS	FFBB	MSXIO, bij de LPTSTT-routine.
H-LSET	FE21	MSXSTS, aan het begin van de LSET-routine.
H-MAIN	FF0C	BINTRP, bij de MAIN-routine.
H-MERG	FE67	SPCDSK, bij de MERGE-routine.
H-MDK$	FE3A	MSXSTS, aan het begin van de MDK$-routine.
H-MKI$	FE30	MSXSTS, aan het begin van de MKI$-routine.
H-MKS$	FE35	MSXSTS, aan het begin van de MSK$-routine.
H-NAME	FDF9	MSXSTS, aan het begin van de NAME-routine.
H-NEWS	FF3E	BINTRP.
H-NMI	FDD6	MSXIO, aan het begin van de NMI-routine.
H-NODE	FEB7	SPCDEV, bij de NODEVN-routine.
H-NOFO	FE58	SPCDSK, bij de NOFOR-routine.
H-NOTR	FF34	BINTRP.
H-NTFL	FE62	SPCDSK, bij de NTFLO-routine.
H-NTFN	FF2F	BINTRP.
H-NTPL	FF6B	BINTRP.

H-NULO	FE5D	SPCDSK, bij de NULOPN-routine.
H-OKNO	FF75	BINTRP.
H-ONGO	FDEA	MSXSTS, aan het begin van de ONGOTP-routine.
H-OUTD	FEE4	BIO, bij de OUTDO-routine.
H-PARD	FEB2	SPCDEV, bij de PARDEV-routine.
H-PHYD	FFA7	MSXIO, bij de PHYDIO-routine.
H-PINL	FDDB	MSXINL, aan het begin van PINLIN.
H-PLAY	FFC5	MSXSTS, bij de aanroep van PLAY.
H-POSD	FEB5	SPCDEV, bij de POSDSK-routine.
H-PRGE	FEF8	BINTRP, bij de PRGEND-routine.
H-PRTF	FF52	BINTRP.
H-PTRG	FFA2	BIPTRG, bij de PTRGET-routine.
H-QINL	FDE0	MSXINL, aan het begin van de QINLIN-routine.
H-READ	FF07	BINTRP, bij de READ-aanroep.
H-RETU	FF4D	MSXSTS, aan het begin van de RSET-routine.
H-RSLF	FE8F	SPCDSK, om de oude drive opnieuw te selecteren.
H-RUNC	FECB	BIMISC, bij de RUNC-routine.
H-SAVD	FE94	SPCDSK, om de huidige drive te bewaren.
H-SAVE	FE6C	SPCDSK, bij de SAVE-routine.
H-SCNE	FF98	BINTRP.
H-SCRE	FFC0	MSXSTS, bij de aanroep van de SCREEN-opdracht.
H-SETF	FE53	SPCDSK, bij de SETFIL-routine.
H-SETS	FDF4	MSXSTS, aan het begin van de SETS-routine.
H-SNGF	FF39	BINTRP.
H-STKE	FEDA	BIMISC, bij de STKERR-routine.
H-TIMI	FD9F	MSXIO, aan het begin van de INTERRUPTIE-routine.
H-TOTE	FDBD	MSXIO, aan het begin van de TOTEXT-routine.
H-TRMN	FF61	BINTRP.
H-WIDT	FF84	BINTRP, bij de WIDTHS-routine.

Sectie 4.9

De systeem-RAM

Dit gedeelte bevat een overzicht van iedere systeemvariabele, voorzien van adres en lengte in bytes. Het adres wordt gegeven in hexadecimale notatie en de lengte in decimale notatie.
De systeem-RAM bevindt zich tussen adres &HF380 en adres &HFFFF. De locaties &HF380 t/m &HF399 bevatten eigenlijk machinecode-routines die worden gebruikt door de slotbesturings-BIOS-routines.

NAAM	ADRES	LENGTE
ARG	F847	16
ARYTA2	F7B5	2
ARYTAB	F6C4	2
ASPCT1	F40B	2
ASPCT2	F40D	2
ASPECT	F931	2
ATRBAS	F928	2
ATRBYT	F3F2	1
AUTFLG	F6AA	1
AUTINC	F6AD	2
AUTLIN	F6AB	2
BAKCLR	F3EA	1
BASROM	FBB1	1
BDRCLR	F3EB	1
BOTTOM	FC48	2
BRDATR	FCB2	1
BUF	F55E	258
BUFEND	FC18	0
BUFMIN	F55D	1
CAPST	FCAB	1
CASPRV	FCB1	1
CENCNT	F933	2
CGPBAS	F924	2
CGPNT	F91F	3
CLIKFL	FBD9	1
CLIKSW	F3DB	1
CLINEF	F935	1
CLMLST	F3B2	1
CLOC	F92A	2
CLPRIM	F38C	14
CMASK	F92C	1
CNPNTS	F936	2
CNSDFG	F3DE	1
CODSAV	FBCC	1
CONLO	F66A	8
CONSAV	F668	1
CONTXT	F666	2
CONTYP	F669	1
CPCNT	F939	2
CPCNT8	F93B	2
CPLOTF	F938	1
CRCSUM	F93D	2
CRTCNT	F3B1	1
CS120	F3FC	10
CSAVEA	F942	2
CSAVEM	F944	1
CSCLXY	F941	1
CSRSW	FCA9	1
CSRX	F3DD	1
CSRY	F3DC	1
CSTCNT	F93F	2
CSTYLE	FCAA	1

CURLIN	F41C	2
CXOFF	F945	2
CYOFF	F947	2
DAC	F7F6	16
DATLIN	F6A3	2
DATPTR	F6C8	2
DECCNT	F7F4	1
DECTM2	F7F2	2
DECTMP	F7F0	2
DEFTBL	F6CA	26
DEVICE	FD99	1
DIMFLG	F662	1
DONUM	F665	1
DORES	F664	1
DOT	F6B5	2
DRWANG	FCBD	1
DRWFLG	FCBB	1
DRWSCL	FCBC	1
DSCTMP	F698	3
ENDBUF	F660	1
ENDFOR	F6A1	2
ENDPRG	F40F	5
ENSTOP	FBB0	1
ERRFLG	F414	1
ERRLIN	F6B3	2
ERRTXT	F6B7	2
ESCCNT	FCA7	1
EXPTBL	FCC1	4
FBUFFR	F7C5	43
FILNAM	F866	11
FILNM2	F871	11
FILTAB	F860	2
FLBMEM	FCAE	1
FLGINP	F6A6	1
FNKFLG	FBCE	10
FNKSTR	F87F	160
FNKSWI	FBCD	1
FORCLR	F3E9	1
FRCNEW	F3F5	1
FRETOP	F69B	2
FSTPOS	FBCA	2
FUNACT	F7BA	2
GETPNT	F3FA	2
GRPACX	FCB7	2
GRPACY	FCB9	2
GRPATR	F3CD	2
GRPCGP	F3CB	2
GRPCOL	F3C9	2
GRPHED	FCA6	1
GRPNAM	F3C7	2
GRPPAT	F3CF	2
GXPOS	FCB3	2
GYPOS	FCB5	2
HEADER	F40A	1
HIGH	F408	2
HIMEM	FC4A	2
HOLD	F83E	8
HOLD2	F836	8
HOLD8	F806	48
INSFLG	FCA8	1
INTCNT	FCA2	2
INTFLG	FC9B	1
INTVAL	FCA0	2
JIFFY	FC9E	2
KANAMD	FCAD	1
KANAST	FCAC	1
KBUF	F41F	318
KEYBUF	FBF0	40
LFPROG	F954	1
LINL32	F3AF	1
LINL40	F3AE	1
LINLEN	F3B0	1
LINTTB	FBB2	24
LINWRK	FC18	40
LOHADR	F94B	2
LOHCNT	F94D	2
LOHDIR	F94A	1
LOHMSK	F949	1
LOW	F406	2
LOWLIM	FCA4	1
LPTPOS	F415	1
MAXDEL	F92F	2
MAXFIL	F85F	1
MAXUPD	F3EC	3
MCLFLG	F958	1
FCLLEN	FB3B	1
MCLPTR	FB3C	2
MCLTAB	F956	2
MEMSIZ	F672	2
MINDEL	F92D	2
MINUPD	F3EF	3
MLTATR	F3D7	2
MLTCGP	F3D5	2
MLTCOL	F3D3	2
MLTNAM	F3D1	2
MLTPAT	F3D9	2
MOVCNT	F951	2
MUSICF	FB3F	1

NAMBAS	F922	2	RS21Q	FAF5	64
NEWKEY	FBE5	11	RTPROG	F955	1
NLONLY	F87C	1	RTYCNT	FC9A	1
NOFUNS	F7B7	1	RUNBNF	FCBE	1
NTMSXP	F417	1	RUNFLG	F866	0
NULBUF	F862	2	SAVEND	F87D	2
OLDKEY	FBDA	11	SAVENT	FCBF	2
OLDLIN	F6BE	2	SAVSP	FB36	2
OLDSCR	FCB0	1	SAVSTK	F6B1	2
OLDTXT	F6C0	2	SAVTXT	F6AF	2
ONEFLG	F6BB	1	SAVVOL	FB39	2
ONELIN	F6B9	2	SCNCNT	F3F6	1
ONGSBF	FBD8	1	SCRMOD	FCAF	1
OPRTYP	F664	0	SKPCNT	F94F	2
PADX	FC9D	1	SLTATR	FCC9	64
PADY	FC9C	1	SLTTBL	FCC5	4
PARM1	F6E8	100	SLTWRK	FD09	128
PARM2	F750	100	STATFL	F3E7	1
PATBAS	F926	2	STKTOP	F674	2
PATWRK	FC40	8	STREND	F6C6	2
PDIREC	F953	1	SUBFLG	F6A5	1
PLYCNT	FB40	1	SWPTMP	F7BC	8
PRMFLG	F7B4	1	T32ATR	F3C3	2
PRMLEN	F6E6	2	T32CGP	F3C1	2
PRMLN2	F74E	2	T32COL	F3BF	2
PRMPRV	F74C	2	T32NAM	F3BD	2
PRMSTK	F6E4	2	T32PAT	F3C5	2
PROCNM	FD89	16	TEMP	F6A7	2
PRSCNT	FB35	1	TEMP2	F6BC	2
PRTFLG	F416	1	TEMP3	F69D	2
PTRFIL	F864	2	TEMP8	F69F	2
PTRFLG	F6A9	1	TEMP9	F7B8	2
PUTPNT	F3F8	2	TEMPPT	F678	2
QUEBAK	F971	4	TEMPST	F67A	30
QUETAB	F959	24	TRCFLG	F7C4	1
QUEUEN	FB3E	1	TRGFLG	F3E8	1
QUEUES	F3F3	2	TRPTBL	FC4C	78
RAWPRT	F418	1	TTYPOS	F661	1
RDPRIM	F380	5	TXTATR	F3B9	2
REPCNT	F3F7	1	TXTCGP	F3B7	2
RG0SAV	F3DF	1	TXTCOL	F3B5	2
RG1SAV	F3E0	1	TXTNAM	F3B3	2
RG2SAV	F3E1	1	TXTPAT	F3BB	2
RG3SAV	F3E2	1	TXTTAB	F676	2
RG4SAV	F3E3	1	USFLG	F6A6	0
RG5SAV	F3E4	1	USRTAB	F39A	20
RG6SAV	F3E5	1	VALTYP	F663	1
RG7SAV	F3E6	1	VARTAB	F6C2	2
RNDX	F857	8	VCBA	FB41	37

VCBB	FB66	37
VCBC	FB8B	37
VLZADR	F419	2
VLZDAT	F41B	1
VOICAQ	F975	128
VOICBQ	F9F5	128
VOICCQ	FA75	128
VOICEN	FB38	1
WINWID	FCA5	1
WRPRIM	F385	7

Het handboek MSX is het meest complete standaardwerk voor MSX-computers. Het is van onschatbare waarde voor zowel de beginnende als ervaren MSX-programmeur.
Om de informatie in dit boek optimaal toegankelijk te maken is gekozen voor een opsplitsing in vier delen. Het eerste deel heeft tot doel de beginnende MSX-programmeur een uitgebreide en heldere uitleg te geven inzake het programmeren in MSX BASIC.
In deel twee worden de meer geavanceerde programmeertechnieken behandeld waarmee de reeds gevorderde programmeur nog meer mogelijkheden krijgt geboden. Tevens komen in dit deel de grafische en geluidsmogelijkheden aan de orde. Het derde deel bevat gedetailleerde uitleg over het programmeren in zowel BASIC als machinetaal.
Tenslotte geeft deel vier een dieper inzicht in de werking van de MSX-computer. Hierin is een volledige gids van het besturingssysteem opgenomen. Voor degene die de mogelijkheden van zijn of haar MSX-computer wil leren kennen is dit handboek onmisbaar.

ISBN 90 201 1876 5